HEARTBEAT | BLENDER | DISTCC | SCTP | RAPIDMIND | APIs

LINUX JOURNAL

Since 1994: The x Community

High-Performance Network Programming

SCTP Multiple Associations

THE PRESENT AND FUTURE KING OF HPC

INTERVIEW WITH RapidMind

HIGH-PERFORMANCE COMPUTING

- RHEL Cluster Suite
- Heartbeat and High Availability
- High-Availability E-Mail with Active Directory
- Distribute Compiles with distcc

NOVEMBER 2007 | ISSUE 163
www.linuxjournal.com

$5.99US $6.99CAN
11
0 74470 03102 4

APPRO

HPC Cluster Solutions

Enterprise and High-Performance Computing Under Your Control

Industry leading 4P x86 computing to boost performance and power efficiency with Dual or Quad-Core AMD Opteron™ Processors

4-Way *Xtreme*Workstation™

- AMD Opteron™ 8000 Series processors
- Up to 128GB of DDR2 533/667 memory
- Up to 6.0TB SATA or 2.4TB SAS
- Hot-swappable drives
- 2 PCI-E x16 slots for high-end graphics card
- Windows® or Linux OS

4-Way 3U *Xtreme*Server™

- AMD Opteron™ 8000 Series processors
- Up to 128GB of DDR2 533/667 memory
- Up to 4.5TB SATA or 1.8TB SAS
- 2 PCI-E x16 and 3 PCI-X slots
- Hot-swappable drives
- Redundant power supplies & fans
- ServerDome Management – IPMI 2.0
- Windows® or Linux OS

For more information, please visit www.appro.com
or call Appro Sales at 800-927-5464

AMD Opteron™ Processors:

- Quad-Core Ready - increase capacity without altering datacenter infrastructure
- Best performance per-watt with energy-efficient DDR2
- Optimized system performance with Direct Connect Architecture

AMD Opteron™ 64

Manage Any Data Center. Anytime. Anywhere.

SEE US AT ANY OF THESE SHOWS!

Infrastructure Mgt. World
Scottsdale, AZ - Booth TBD
Sept. 10 - 12

VM World
San Francisco, CA - Booth 1113
Sept. 11 - 13

AFCOM Data Center World
Dallas, TX - Booth 436
Sept. 17 - 18

High Perf. on Wall Street
New York, NY - Booth 216
Sept. 17

IDC Enterprise Infra. Forum
New York, NY - Booth TBD
Sept. 20

Interface Salt Lake City
Salt Lake City, UT - Booth 309
Oct. 4

GEOINT
San Antonio, TX - Booth 374
Oct. 22 - 24

Interop New York Fall
New York, NY - Booth 543
Oct. 24 - 25

AFCEA Asia-PAC TechNet
Honolulu, HI - Booth 516
Nov. 4 - 9

Super Computing
Reno, NV - Booth 164
Nov. 12 - 15

LISA
Dallas, TX - Booth 200
Nov. 14 - 15

DaCEY Awards
Atlanta, GA
Nov. 15

Gartner Data Center Conf.
Las Vegas, NV - Booth TBD
Nov. 27 - 30

Interface Seattle
Seattle, WA - Booth 206
Nov. 28

Avocent builds hardware and software to access, manage and control any IT asset in your data center, online or offline, keeping it, and your business, "always on".

Visit us on our Remote Control Tour. For locations near you, go to **www.avocent.com/remotecontrol.**

Avocent®
The Power of Being There®

CONTENTS

NOVEMBER 2007
Issue 163

HIGH-PERFORMANCE COMPUTING

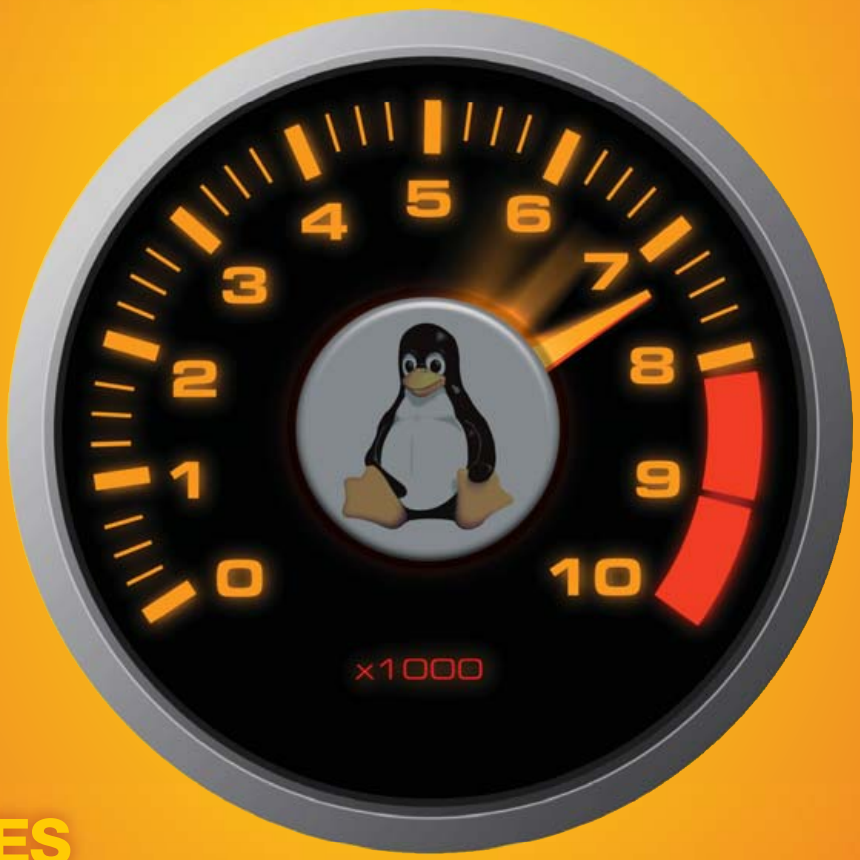


IMAGE ©ISTOCKPHOTO.COM/NARVIKK

FEATURES

ON THE COVER

Agility.

With Coyote Point, you'll never have to wonder if your network is fast enough or flexible enough. You'll know it is.

From local to global load balancing, application acceleration or ultimate network manageability, Coyote Point leads the pack. We take the guesswork and difficulty out of application traffic management. You won't find anything faster, smarter or more affordable.

Find out why more than 2,000 businesses rely on us to maximize their infrastructure. Learn what Coyote Point could mean to your Web and network traffic.

Write info@coyotepoint.com or call 1-877-367-2696.

Coyotes respond with lightning speed. Your network should, too.

CONTENTS NOVEMBER 2007 Issue 163

COLUMNS

IN EVERY ISSUE

INDEPTH

Next Month

LAPTOPS

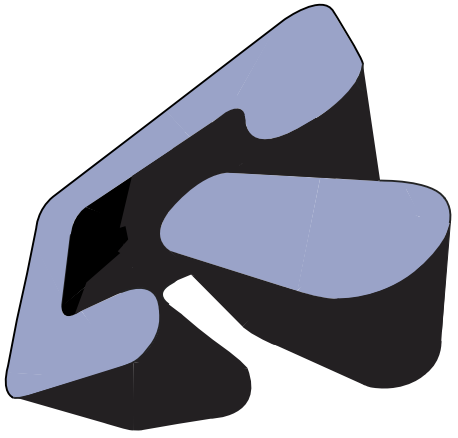
It has been a good year for innovation in Linux laptops with many new models, technologies and brands now at your disposal. In next issue's Laptop Buying Guide, we'll tell you what to look for and where to buy the Linux laptop that best fulfills your needs and won't bust your budget.

As always, there's much more. Reuven M. Lerner will show you how to create your own Facebook applications, and Ben Martin will explain how to protect your laptop files with FUSE and an on-line storage service.

USPS *LINUX JOURNAL* (ISSN 1075-3583) is published monthly by Belltown Media, Inc., 2211 Norfolk, Ste 514, Houston, TX 77098 USA. Periodicals postage paid at Houston, Texas and at additional mailing offices. Cover price is $5.99 US. Subscription rate is $25/year in the United States, $32 in Canada and Mexico, $62 elsewhere. POSTMASTER: Please send address changes to *Linux Journal*, PO Box 980985, Houston, TX 77098. Subscriptions start with the next issue.

All your cores are belong to us.

80 Cores starting at $33,900



QSOL.COM
Server Appliances
800.933.7510

LINUX JOURNAL

Editor in Chief
Nick Petreley, ljeditor@linuxjournal.com

Executive Editor	Jill Franklin jill@linuxjournal.com
Senior Editor	Doc Searls doc@linuxjournal.com
Art Director	Garrick Antikajian garrick@linuxjournal.com
Products Editor	James Gray newproducts@linuxjournal.com
Editor Emeritus	Don Marti dmarti@linuxjournal.com
Technical Editor	Michael Baxter mab@cruzio.com
Senior Columnist	Reuven Lerner reuven@lerner.co.il
Chef Français	Marcel Gagné mggagne@salmar.com
Security Editor	Mick Bauer mick@visi.com

Contributing Editors
David A. Bandel • Ibrahim Haddad • Robert Love • Zack Brown • Dave Phillips
Marco Fioretti • Ludovic Marcotte • Paul Barry • Paul McKenney • Dave Taylor

Proofreader Geri Gale

Publisher	Carlie Fairchild publisher@linuxjournal.com
General Manager	Rebecca Cassity rebecca@linuxjournal.com
Director of Sales	Laura Whiteman laura@linuxjournal.com
Regional Sales Manager	Joseph Krack joseph@linuxjournal.com
Regional Sales Manager	Kathleen Boyle kathleen@linuxjournal.com
Circulation Director	Mark Irgang mark@linuxjournal.com
Marketing Coordinator	Lana Newlander mktg@linuxjournal.com
System Administrator	Mitch Frazier sysadm@linuxjournal.com
Webmaster	Katherine Druckman webmaster@linuxjournal.com
Accountant	Candy Beauchamp acct@linuxjournal.com

Linux Journal **is published by, and is a registered trade name of, Belltown Media, Inc.**
PO Box 980985, Houston, TX 77098 USA

Subscriptions
E-MAIL: subs@linuxjournal.com
URL: www.linuxjournal.com/subscribe
PHONE: +1 713-589-3503
FAX: +1 713-589-2677
TOLL-FREE: 1-888-66-LINUX
MAIL: PO Box 980985, Houston, TX 77098 USA
Please allow 4–6 weeks for processing address changes and orders
PRINTED IN USA

Power of 32 Processors

256GB of Memory

Based on industry leading AMD Opteron™ microprocessors, the **HPC A5808-32** server provides industry leading scalable x64 processing with the latest I/O, network, memory, and power efficient technologies.

With features such as **AMD quad-core Opteron™** microprocessors and up to 256GB of DDR2 for memory-intensive performance, multithreaded applications like CRM, ERP, e-commerce, and virtualization will see significant performance improvements.

And **HPC MasterSight™** delivers features to help manage the system with inclusive diagnostic tools.

Specializing in High Performance Computers, HPC systems, Inc. is a solution provider serving the financial, health, educational, and government sector. We appreciate that you have a choice of dozens of vendors, but not all of them have our uncompromising dedication and experience in producing the best solutions.

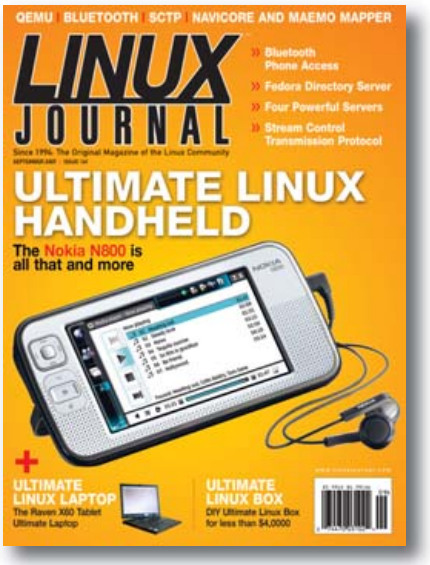
HPC Systems, Inc.
48009 Fremont Blvd
Fremont, CA 94538
Tel: 510-656-8282
Fax: 510-656-8341
E-mail: info@hpcsystems.com

AMD Solution Provider PLATINUM

SBA 8(A) Certified

GSA GS-35F-0596R Contract Holder

letters



Your Wish Is Our Command

I would love to see a detailed explanation on the new Completely Fair Scheduler in the Linux kernel, including areas where it may be inferior to the existing scheduler and where it excels. Also, I'd like to see an explanation of the tickless patches, which apparently result in significant power savings.

--

Chris Thompson

We'll look into a scheduling article, but we have an article on power savings in the works, including information on the tickless patches.—Ed.

The Arcade Game Machine Is Hot!

I was very interested to read Shawn Powers' article on building an arcade machine based around Linux [*LJ*, August 2007]. However, one thing that concerns me is the lack of reference to cooling issues. If a PC is in a wooden box for any period of time without a fan to force cool air through, there would be a risk of heat damage to the machine, particularly the CPU and the hard drive (**www.coker.com.au/energy/computer-power.html**).

The above URL has the energy use of some computers I have owned at various times (the idle tests were idle at a Linux prompt—idle at a DOS prompt or the BIOS uses significantly more power). You will note that a P4 machine uses significantly more power than a PIII. I can only guess at the relative power requirements of a PIII and the original game hardware, but I expect that all PCs manufactured in the last ten years use more power and produce more waste heat. Even a PIII might have cooling problems!

One thing that is not widely known is that a P4 will generally take more clock cycles than a PIII to complete a given amount of work—a 1.5GHz P4 will often deliver the same performance as a 1GHz PIII! If a machine is to run in a poorly ventilated space, a PIII is a significantly better option than a P4.

Hard drives take as much as 5W when idling. Use of other mass storage devices (such as USB) might reduce the power use for such a machine.

I have a small stockpile of old PIII machines for occasions when I need a low-power machine.

--

Russell Coker

The Inevitable Descent into Anarchy

The Linux OS is the most glamorous (I'm serious) example of [the] trend [toward data anarchy], but Linux is only an outgrowth of a much bigger universe of collaborative software development that uses open-source licenses and techniques to leverage the talents and creative impulses of hundreds of thousands of people worldwide. Cheap MP3 players and P2P file-sharing networks are another.

Direct evidence of the trend is plummeting CD sales as users swap CDs, copy new ones, file share and port their files to any device they want. The efforts of the RIAA are becoming laughable, as they sue a few unfortunates and scold the rest of the population from a position of pious hypocrisy. They are the biggest thieves of all. Also pathetic are the many pop stars, now grown rich and fat sucking on the corporate tit, who parrot the RIAA's line. It matters not.

The only reason artists and record companies ever had control over music media was that it was impractical for listeners to create their own. Given that ability, most listeners will follow my own common-sense creed: when I get the data, it's my data, and I'll do with it what I damn well please.

The RIAA hands down morality lessons while cracking the lawsuit whip in hope of putting the file-sharing genie back in its bottle. They pretend that there is some moral code that forbids consumers to copy and give away music media. Their morality is based on laws that were manufactured by a Congress eager to be bribed. It's easy to burst the RIAA's moral bubble. Just travel back to 1850 in your mind and go hear a music performer.

In 19th-century America, the artists played and the people enjoyed. It had been that way for 10,000 years. If the artists were lucky or very good, they could make a living by performing. Many do that to this very day. It is an honest living. They work at what they love. It's their job.

It was a temporary quirk of technology that allowed artists and record companies to turn performances into commodities to be bought and sold. A few lucky artists could perform, record and then sit back and let the bucks roll in while they snorted coke and bought mansions. That's OK with me, and it still works for some. But, the data is written on the wall, and it is an age-old warning that applies to all: adapt or die.

Readers may wonder what all this has to do with cell-phone networks. Carl Brown suggested that there is nothing fundamentally blocking the concept of a user-created cell-phone network. It also is true that there is nothing blocking users from building their own worldwide data network, uncontrolled by governments or corporations. If I can share my Wi-Fi with the neighbors in my apartment, I can bridge to the guy across the street who can run a directional shot to his buddy a quarter-mile away who can...(ad infinitum).

The idea was not invented by me, although I humbly declare I thought of it myself. The idea is so obvious, anyone with an understanding of networks is likely to conceive of it. A quick Google search brings to light that user-operated and user-supported community networks are already providing free service to many people. The ISPs don't like it, but there's nothing they can do about it. Those simple networks, however, are only attachments to the Internet. They still depend on the corporate infrastructure.

Another and related sign of the trend toward anarchy is revealed in the project called Netsukuku (**netsukuku.freaknet.org**). That project community has created a dæmon that uses network interfaces to communicate directly with the same dæmon in a connected

servers DIRECT™
MORE PRODUCTS, BETTER SERVICE, GUARANTEED.

GO STRAIGHT TO THE SOURCE!

KEEP YOUR BUSINESS RUNNING SMOOTHLY

PROTECT YOUR SMALL BUSINESS WITH THE BUILT-IN SECURITY ENHANCEMENTS OF THE DUAL-CORE INTEL® XEON® PROCESSOR IN YOUR SERVERSDIRECT SYSTEM.

SDR-1105T 1U ENTRY LEVEL SERVER

$1,129

EXCELLENT GENERAL PURPOSE SERVER FOR ORGANIZATIONS WITH THE NEED FOR A LOW, ENTRY LEVEL PRICE

* 1U Rackmount Chassis with 520W power supply
* Supermicro X7DVL-L Server Board with Intel® 5000V (Blackford VS) Chipset
* Intel Quad-Core Xeon Processor E5310 1.6GHZ
* Total 512MB, 2pcs x 256MB Kingston DDR2 533Mhz FB-DIMM ECC
* Seagate SATAII 160GB 7200 RPM 8MB Cache SATA 3.0Gb/s Hard Drive
* 4 x 1" Hot-swap SATA Drive Bays
* Two Intel® 82563EB Dual-port Gigabit Ethernet Controller
* Intel® ESB2 SATA 3.0Gbps Controller RAID 0, 1, 5, 10 support

STARTING PRICE **$1,259**

SDR-2503T 2U APPLICATION SERVER

Highest performing with Dual Core/ Quad Core Xeon CPU based. Excellent with general purpose applications and provide the most power.

* 2U Rackmount Chassis with 650W power supply
* Supermicro X7DVL-E Server Board with Intel® 5000V (Blackford VS) Chipset
* Intel Quad-Core Xeon Processor E5310 1.6GHZ
* Total 512MB, 2pcs x 256MB Kingston DDR2 533Mhz FB-DIMM ECC
* Seagate SATAII 250GB 7200 RPM 16MB Cache SATA 3.0Gb/s Hard Drive
* 6 x 1" Hot-swap SATA Drive Bays
* Intel® (ESB2/Gilgal) 82563EB Dual-port Gigabit Ethernet Controller
* Intel® ESB2 SATA 3.0Gbps Controller RAID 0, 1, 5, 10 support

STARTING PRICE **$1,999**

SDR-3500T 3U DATABASE SERVER

Easily Scalable storage solution with hot-swap functionality for growing businesses

* 3U Rackmount chassis with Redundant 800W power supply
* Supermicro X7DBE+ Server Board with Intel® 5000P (Blackford) Chipset
* Intel Quad-Core Xeon Processor E5310 1.6GHZ
* Total 1024MB, 2pcs x 512MB Kingston DDR2 533MHz FB-DIMM ECC
* Seagate SATAII 500GB 7200 RPM 16MB Cache SATA 3.0Gb/s Hard Drive
* 16 x 1" Hot-swap SATA Drive Bays
* Dual-port Gigabit Ethernet Controller
* Intel ESB2 SATA 3.0Gbps Controller RAID 0, 1, 10 support

STARTING PRICE **$1,199**

SDR-7045B-TB 4U FILE SERVER

4U Quad-Core Xeon Server offers excellent value and expandability

* 4U Rackmountable / Tower with 650W power supply
* Supermicro Super X7DBE Server Board with Intel® 5000P (Blackford) Chipset
* Intel Quad-Core Xeon Processor E5310 1.6GHZ
* Total 1024MB, 2pcs x 512MB Kingston DDR2 667MHz FB-DIMM ECC
* Seagate SATAII 750GB 7200 RPM 16MB Cache SATA 3.0Gb/s Hard Drive
* 6 x 3.5" Hot-swap SAS/SATA Drive Bays
* Dual-port Gigabit Ethernet Controller
* Intel ESB2 SATA 3.0Gbps Controller RAID 0, 1, 5, 10 support

STARTING PRICE **$3,099**

SDR-5111T 5U ADVANCED STORAGE SERVER

Quad Core dual Xeon CPU based, with 24 hot-swap hard disk bays suitable for 18TB of pure data Storage capacity

* 5U Rackmount chassis with Redundant 1350W power supply
* Supermicro X7DBE Server Board with Intel® 5000P (Blackford) Chipset
* Intel Quad-Core Xeon Processor E5310 1.6GHZ
* Total 1024MB, 2pcs x 512MB Kingston DDR2 667MHz FB-DIMM ECC
* Seagate 750GB 7200 RPM 16MB Cache SATA 3.0Gb/s Hard Drive
* 24 x 1" Hot-swap Drive Bays
* Intel® (ESB2/Gilgal) 82563EB Dual-port Gigabit Ethernet Controller
* Intel ESB2 SATA 3.0Gbps Controller RAID 0, 1, 5, 10 support

SERVERS DIRECT CAN HELP YOU CONFIGURE YOUR NEXT HIGH PERFORMANCE SERVER SYSTEM - CALL US TODAY!

Our flexible on-line products configurator allows you to source a custom solution, or call and our product experts are standing by to help you assemble systems that require a little extra. Servers Direct - your direct source for scalable, cost effective server solutions.

1.877.727.7887 | www.ServersDirect.com

intel Core 2 Quad inside
Quad-core. Unmatched.

computer. In theory, the mesh network so created can mimic the Internet's TCP/IP layer with a decentralized domain name system that uses distributed architecture and builds routes that connect the computers attached to its network. It is, as the creators say, anarchical.

I do not claim to see the future. I only extrapolate what seems to be an inexorable transfer of communications and data management power from the elite to the masses. If you believe in people, this can only be a good thing. If you fear people, or begrudge them control over their own lives, then you will fight it...and lose.

--
Tim Copeland

Practical Linux Box

I eagerly awaited the current issue of *LJ* [September 2007] to read the Ultimate Linux Box article. I plan to build a system very soon and hoped to get some ideas.

Unfortunately, I just skimmed the article and was very disappointed. What I really would like to see is a Practical Linux Box. No, your sidebar "Penultimate" doesn't address my needs any more than the ULB. Your ULB is really the Ultimate [Windows] Gaming Box, isn't it? For example, look at the Display Card section. It's the latest and greatest DirectX 10 card? How is this a good Linux box?

The things I'm looking for may not be what everyone else is looking for, but I'd like to think there are enough people to warrant some practical thinking in this type of article, such as:

- Quiet/fanless power supply and case.
- Small case with enough room for DVD and two hard drives.
- Affordable: $2,000–$4,000 is not practical or affordable.
- Onboard video is ok.

You could argue that I want a media PC, but that's not really my goal. I'd settle for one. I'm estimating that I can build a system that fits the above specs for about $700, without monitor. Other names for what I want might be the Quiet Linux Box or the Affordable Linux Box.

--
JT Moree

The Ultimate Linux Box had nothing to do with Windows games or serving up DirectX 10, which isn't even very useful on Windows yet. We'll consider your desires for the next ULB issue though.—Ed.

Protection Money

I have to congratulate you on your /var/opinion "The Benevolent Racketeer" [*LJ*, August 2007].

Your imaginary letter writer speaks right out of the soul of everybody. The crowd reading the lines will equally agree and see their points addressed as the crowd reading between the lines. Luckily, I paid my insurance money to *Linux Journal*, so you will not sue me for reading this terrific piece of word code.

--
Kurt

mkdir Errors

I too have a small nit to pick with respect to sample code and the lack of error handling. It is trivial to add at least some minimal error handling. In the September 2007 issue's Letters, Jack points out that mkdir error handling is non-existent. Bash scripts are notorious for not doing any error handling. As for the mkdir issue that Jack points out, I would suggest the following:

```
mkdir -p $dimension
if [ $? -ne 0 ]; then
        echo "Error: could not create\
        directory $dimension"
        return 1
fi
```

The -p switch to mkdir will create all directories in the $dimension string recursively and will not complain if any or all directories already exist. The main reason for failure will be user permissions. This is handled by echoing the error and returning an error value to the caller.

If the goal of these sample scripts is to instruct beginners to learn techniques with bash, error handling cannot be ignored. The same goes for Ruby, Python, Perl, C and C++. The code samples in the SCTP (why does that make me think of John Candy) article are excellent in this regard.

--
steeve

Maximize **AMD Opteron™ Quad-Core** Performance with **UniServer**

Up to 32GB memory on 8 DIMM sockets

1U UniServer with 1 Socket

- Dual/Quad-Core AMD Opteron™ 2000 series
- Up to 32GB of memory
- 2 hot swap SATA 3.5" HDDs
- MCP55V Pro chipset
- Slim CD/DVD-ROM
- 1 PCI-Express x16 slot
- Remote management-IPMI 2.0

Find Full line of Innovative AMD Opteron™ Servers and Workstation Platforms

1U Server: 1 & 2 Sockets, up to 64GB memory
2U Server: 2 & 4 Sockets, up to 128GB memory
3U Server: 2 & 4 Sockets, up to 128GB memory
Workstation: 1, 2 & 4 Sockets, up to 128GB memory

BELL MICRO www.bellmicro.com 1-800-291-2070

AVNET Applied Computing Solutions www.avnet.com 1-888-300-8277

SYNNEX CORPORATION www.synnex.com 1-888-756-4888

UNIWIDE Technologies

www.**uniwide**.com

1-877-520-0071

Uniwide is an official AMD Validated Server Program Partner

AMD 64 Opteron™

diff -u
WHAT'S NEW IN KERNEL DEVELOPMENT

Linus Torvalds has expressed keen interest in finding someone to put together a full **git repository** of the kernel, going all the way back to version 0.01. He's tried this himself a couple times, and other folks have made various efforts, but it's a hard problem. Certainly, it would not be possible to include every patch that went into the kernel, in the order it was included, because many patches never were sent to any public forum. Even finding the release announcements for all the numbered versions will be difficult, and some official releases are thought to be lost as well. It's a daunting task, but a very valuable one, even if done incompletely. If someone can do it, Linus has offered to comment the various early patches and releases, from memory.

Mingming Cao has submitted patches to allow the **ext4 filesystem** to perform **checksum** calculations on its journal to make sure that any corruption is identified as quickly as possible. With interest from various folks, including **Andrew Morton**, it looks like this feature quickly will be adopted into the official tree, although ext4 still remains a fairly experimental filesystem.

LinuxConf Europe 2007 (LCE) will host a semi-formal discussion of **containers within the kernel**. **Serge E. Hallyn** recently announced plans to arrange for a conference room (including phone lines for anyone who can't be present but still wants to participate) and a series of half-hour presentations. Containers provide a way to cluster processes into specific namespaces that are isolated from the rest of the system and are related to virtualization projects like **Xen**.

Michal Piotrowski has announced the **"Linux Kernel Tester's Guide"**, translated by **Rafael J. Wysocki**, at **www.stardust.webpages.pl/files/handbook/handbook-en-0.3-rc1.pdf**. It is a long document representing much work, it reads like a book, and it clearly explains a lot of material that most discussions on the linux-kernel mailing list tend to assume—for example, how to do a binary search with git to identify precisely when a particular bug was introduced into the tree.

Several projects have changed hands recently. **Valerie Henson** has had to abandon the **Tulip driver**, and now it looks like **Kyle McMartin** may become the official maintainer. **Wim Van Sebroeck** has submitted a patch to make **Mike Frysinger** the official maintainer of the **Blackfin Watchdog driver**. **Mike Sharkey** of **Pike Aerospace Research Corporation** has volunteered to take over the otherwise unmaintained **Parallel Port driver** on behalf of his company. And, **Anton Vorontsov** recently became a co-maintainer of the **Power Supply subsystem**, along with **David Woodhouse**.

Over time, various features have gone into the kernel to support more and more modern architectures. But, for some of these features that have no serious negative impact on older hardware, such as the 386 processor, there's been no real effort to isolate the unneeded features from that older hardware. Kernels compiled for those systems, therefore, have tended to have larger and larger binaries and to require more and more RAM to run. For the most part, no one notices or cares, because most people don't bother running Linux on the 386 anymore. But, the effect has been there, building gradually.

Jonathan Campbell recently started submitting patches to ensure that architectures like the 386 would compile only features that actually would work on those systems. So, things like the **Pentium TSC register** would not be included in compiled 386 kernel binaries. The result of his work was a much smaller binary, and his patches probably will be adopted into the main kernel tree. This kind of support for legacy systems might have an impact on projects to bring computing resources to third-world countries and impoverished neighborhoods or lower the cost of experimenting with clustered solutions.

—ZACK BROWN

LJ Index, November 2007

1. Average measured speed in MBps of a broadband connection with "up to 8Mbps" download speed: 2.7
2. Lowest measured speed in KBps of a broadband connection with "up to 8Mbps" download speed: 90
3. Number of consumers out of five who get the broadband speed they signed up for: 1
4. Percent of surveyed consumers who have felt misled by providers' advertising: 30
5. Billions of Internet users in 2006: 1.1
6. Additional millions of Internet users expected by 2010: 500
7. Millions of video streams per day served by YouTube: 100
8. Number of surveillance cameras in London: 200
9. Trillions of bits sent by London surveillance cameras to their data center: 64
10. Terabytes accumulated per day by Chevron: 2
11. Total exabytes of data in 2006: 161
12. Multiple in millions of 2006 data total to all information in all books ever written: 3
13. Percentage of the digital universe that will be created by individuals by 2010: 70
14. Percentage of the current digital universe that is subject to compliance rules and standards: 20
15. Percentage of the current digital universe that is potentially subject to security applications: 30
16. Exabytes of "user-generated content" expected by 2010: 692
17. Total exabytes of data expected by 2010: 988
18. Percentage of the 2010 digital universe for which organizations will be responsible for security, privacy, reliability and compliance: 85
19. Exabyte capacity of media ready to store newly created and replicated data in the 2010 digital universe: 601
20. Year in which the amount of information created will surpass available storage capacity for the first time: 2007

Sources: 1, 2: which.co.uk, sourced by David Meyer for ZDNet UK | 3, 4: moneysupermarket.com, sourced by David Meyer for ZDNet UK | 5–20: "Expanding the Digital Universe", by John F. Gantz, et al., a March 2007 IDC whitepaper

EmperorLinux

...where Linux & laptops converge

Portable

Since 1999, EmperorLinux has provided pre-installed Linux laptops to universities, corporations, government labs, and individual Linux enthusiasts. Our laptops range from full-featured ultra-portables to desktop replacements. All systems come with one year of Linux technical support by phone and e-mail, and full manufacturers' warranties apply.

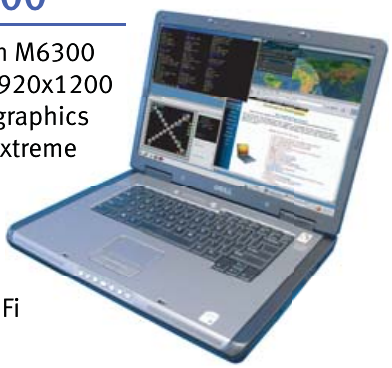
Toucan T61/T61ws

ThinkPad T61/T61ws by Lenovo
- Up to 15.4" WUXGA w/ X@1920x1200
- NVidia Quadro NVS 140M graphics
- 1.8–2.4 GHz Core 2 Duo
- 512 MB–4 GB RAM
- 80–160 GB hard drive
- CDRW/DVD or DVD±RW
- 5.2–6.0 pounds
- 10/100/1000 Mbps ethernet
- 802.11a/b/g (54Mbps) WiFi
- Starts at $1650

Powerful

EmperorLinux specializes in the installation of Linux on a wide range of the finest laptops made by IBM, Lenovo, Dell, Sony, and Panasonic. We customize your choice of Linux distribution to your laptop and provide support for: ethernet, wireless, X-server, ACPI power management, USB, EVDO, PCMCIA, FireWire, CD/DVD/CDRW, sound, and more.

Rhino D830/M6300



Dell Latitude D830/Precision M6300
- Up to 17" WUXGA w/ X@1920x1200
- NVidia Quadro FX 3500M graphics
- 1.8–2.4 GHz Core 2 Duo/Extreme
- 512 MB–4 GB RAM
- 60–160 GB hard drive
- DVD±RW or Blu-ray
- 6.3–8.6 pounds
- 802.11a/b/g (54Mbps) WiFi
- ExpressCard/EVDO
- Starts at $1365

Unique

EmperorLinux offers Linux laptops with unique features. Ruggedized Panasonic laptops are designed for harsh environments: drops, vibrations, sand, rain, and other extremes. ThinkPad tablet PCs are like other laptops, with an LCD digitizer for pen-based input both as a mouse and with pressure sensitivity for writing and drawing on-screen.

Raven X61 Tablet

ThinkPad X61 Tablet by Lenovo
- 12.1" SXGA+ w/ X@1400x1050
- 1.6 GHz Core 2 Duo
- 1–4 GB RAM
- 80–120 GB hard drive
- 3.8 pounds
- Pen/stylus input to screen
- Dynamic screen rotation
- Handwriting recognition
- X61s laptops available
- Starts at $2300

www.EmperorLinux.com 1-888-651-6686

From ACCESS to Excess

The first shoe dropped in September 2005, when ACCESS Co. Ltd. of Japan announced that it would acquire Palm OS developer PalmSource for $324 million. The next shoe dropped in February 2006, when PalmSource detailed the ACCESS Linux Platform (ALP), as an environment open to running Palm OS binaries and Java applications, in addition to native Linux apps. Enough other shoes have dropped since then to give the clear message that ALP has legs. The latest was at LinuxWorld Expo in August 2007, when the company showed screenshots of an iPhone-like UI and provided more details around its plan to make Linux the most supportive environment for mobile device and application development. Chief among these is the Hiker Application framework that fills in some of the formerly missing APIs for mobile applications. Hiker originally was available through the MPL (Mozilla Public License), but it reportedly will be dual-licensed with the LGPL (v2) license as well.

Looking beyond the superficial resemblances between what ACCESS showed and the now-familiar iPhone, it's clear that the mobile application market will divide between Web-based (iPhone) and native OS-based (ACCESS, OpenMoko, maemo)—with the latter embracing legacy apps developed for other platforms as well. Thus, the pavement gets wider on the road to truly open mobile devices and markets that grow on them.

—DOC SEARLS

Raising the Social Tide

Brad Fitzpatrick has a long pedigree for a young guy. Besides creating LiveJournal—one of the earliest and most popular blogging services (and an open-source one at that)—he is the creator of OpenID, Perlbal, MogileFS, memcached, djabberd and many other fine hacks. Astute readers may recall "Distributed Caching with Memcached", which Brad wrote for *Linux Journal* in 2004. Today, memcached is one of the world's most widely used distributed caching methods, while OpenID is a breakaway leader in the user-centric identity field.

In August 2007, Brad published "Thoughts on the Social Graph" (**bradfitz.com/social-graph-problem**), which presents this problem statement:

> There are an increasing number of new "social applications" as well as traditional applications that either require the "social graph" or that could provide better value to users by utilizing information in the social graph. What I mean by "social graph" is the global mapping of everybody and how they're related, as Wikipedia describes it (**en.wikipedia.org/wiki/Social_graph**). Unfortunately, there doesn't exist a single social graph (or even multiple graphs that interoperate) that's comprehensive and decentralized. Rather, there exists hundreds of disperse social graphs, most of dubious quality and many of them walled gardens.

At the time, Wikipedia's "social network" entry (same as its "social graph" entry) said, "Social network analysis views social relationships in terms of nodes and ties. Nodes are the individual actors within the networks, and ties are the relationships between the actors." Elsewhere in Wikipedia, "Graph" is explained as a mathematical concept, "a set of objects called points, nodes, or vertices connected by links called lines or edges".

So, the first idea is to move the center of social networking gravity outside the silo'd sites, each of which impose inconveniences on themselves as well as their members. As Brad says, "People are getting sick of registering and re-declaring their friends on every site." The process is also silly. Not long after Google's Orkut social network site went up, Rael Dornfest made fun of the friendship-declaring protocol by walking up to people he knew and saying, "You are my friend? Yes or no?"

The second idea is to make society itself a platform. Or, in Brad's terms, to "make the social graph a community asset", and "to build the guts that allow a thousand new social applications to bloom".

Significantly, most social network sites (all but MySpace, I believe) run on Linux. Wikipedia too. If Brad's right, we can equip the rising social network tide that lifts all boats. There should even be plenty of work converting today's silos into tomorrow's arks.

For more, visit **bradfitz.com/social-graph-problem**, or just look up "social graph" on the vast Linux hack called Google. The lucky top result probably will be Brad's.

—DOC SEARLS

Bring SMS to the Live Web with a FoxBox

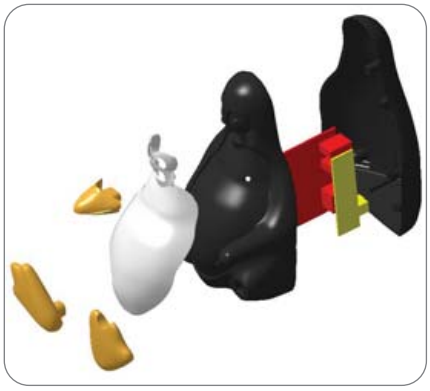
Acme's SMS FoxBox

The forces at Acme Systems and KDev, two veteran embedded Linux system and software developers based in Italy, have conspired to bring the world SMS FoxBox. It's a Linux-based box dedicated to sending and receiving SMS messages that can be managed through a Web interface. It can handle up to 30 incoming messages at a time on a common SIM card. It also works as an SMS to TCP/IP gateway, so you can combine SMS messaging with network and user applications. You can SMS to and from e-mail, MySQL, Web scripts, desktop widgets, whatever.

SMS FoxBox is ideal for use in Live Web conditions. Broadcasters can use it to interact with their audiences. Emergency services can use it to flow live reports onto Web sites or broadcasts. Lightweight monitoring, alarm sending, trouble ticketing and remote device management can be moved to SMS from other methods. Databases and address books can be kept current.

The unit comes with a GSM quad-band modem, an SD/MMC card for storing messages (it comes default with a 512MB card). It runs on the 2.6 Linux kernel, has a BIS module for failover to up to two backup appliances and an internal clock with a backup battery and NTP support.

By the way, you might remember Acme as the source not only of the tiny Acme Fox embedded Linux system board, but for its optional Tux Case as well. Today, that also can come with an Acme Fox SBC (single-board computer). With Kdev's FoxServe firmware, it can work as a dynamic Web server (Apache, PHP, SQLite, SSL/TLS and so on).

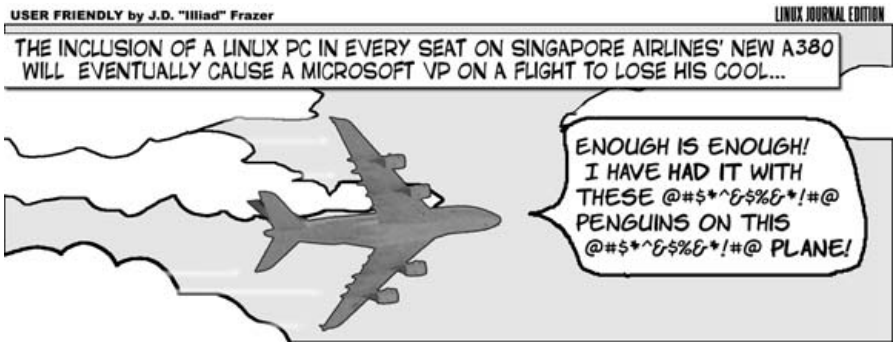


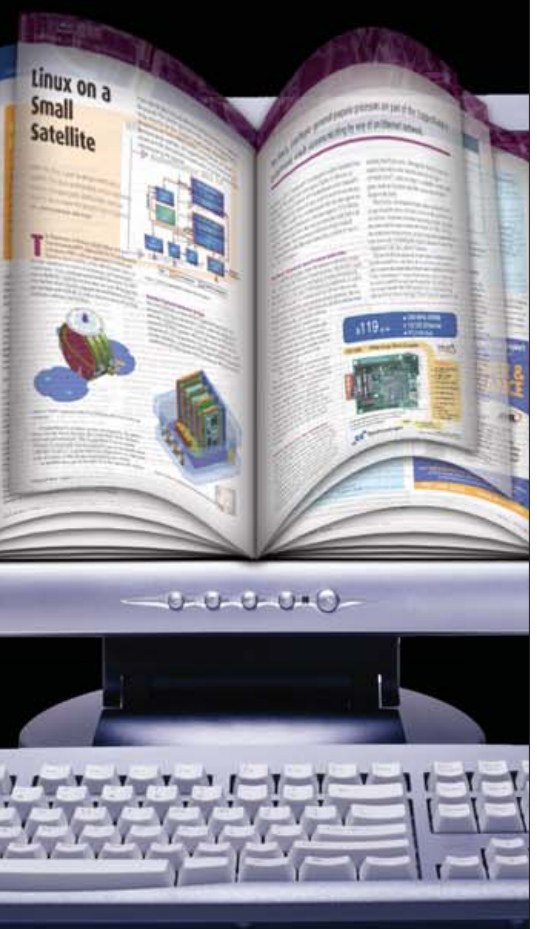
Acme's Tux Case (Showing Separate Parts)

Resources

Acme Systems: **www.acmesystems.it**
KDev: **www.kdev.it/joomla**
Company Visuals: **www.kdev.it/HTDOCS/press.html**

—DOC SEARLS





Frisky N800 Hack

Keijo Lähetkangas has a fun job at Nokia: taking the company's N800 handheld computer (subject of our September 2007 cover story) and turning it into the brains, faces and controllers of Robot pets. His Puppy and a four-wheeled Rover companion were stars in the demo room at this year's Guadec conference in Birmingham (UK). Puppy is remarkably flexible and expressive. It walks, sits, smiles, dances, sleeps and even—how can we put this politely?—lifts one leg (meaning that Puppy is a male, or at least acts like one).

Several Guadec attendees compared Puppy with Sony's Aibo, the most notable difference (besides looks) being the Puppy's openness. It's a pure DIY pet—make of him what you like. Also, Aibo had no eyes. Puppy not only has eyes on his display, but he also can look at the world through the N800's built-in camera, which is in a barely noticeable black cylinder that pops out the side of the N800. It got confusing to take pictures of Puppy and Rover while they took pictures of me, which appeared on other N800s over wireless connections.

Puppy and Rover's non-N800 hardware is all from the Robotics.com catalog. The controlling software, all open source, is at **https://garage.maemo.org/projects/robot**. You also can see Puppy in action at **youtube.com/puppyrobot**.

—DOC SEARLS

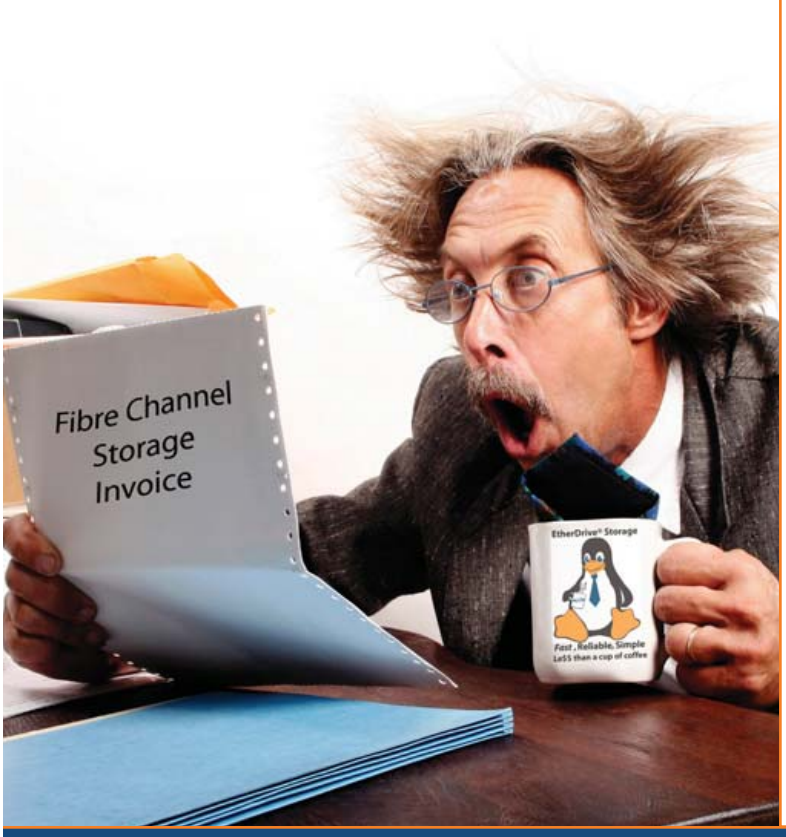
They Said It

If our customers buy bandwidth from us, and they want to share it with neighbors, or publicly, that doesn't make them bad customers.
—Joe Plotkin, Bway.net, www.businessweek.com/technology/content/jul2002/tc2002073_1130.htm

---

...the commercial concerns from the very beginning, even when they were small, were really very important. The commercial distributions were what drove a lot of the nice installers, and pushed people to improve usability....I think commercial users of Linux have been very important in actually improving the product. I think all the technical people who have been involved have been hugely important, but I think that the kind of commercial use that you can get with the GPLv2 is also important—you need a balance between pure technology, and the kinds of pressures you get from users through the market....If you have a purely marketing (or customer) driven approach, you end up with crap technology in the end. But I think that something that is purely driven by technical people will also end up as crap technology in the end, and you really need a balance here.
—Linus Torvalds, www.linuxworld.com/news/2007/080907-torvalds-on-linux-ms-softwares.html?page=2

It's exciting to go to work each day knowing that scads of companies are using your software, then contacting you to get additional value. It's not easy by any stretch, but it's a lot more efficient and productive than the proprietary model.
—Matt Asay, blogs.cnet.com/8301-13505_1-9762210-16.html?part=rss&tag=feed&subj=TheOpenRoad



Are you Shocked

by the high cost of iSCSI & Fibre Channel SAN storage?

AoE is the answer!

ATA-over-Ethernet = ***Fast, Reliable, Simple*** storage.

www.coraid.com

EtherDrive® SRxxxx

- Fast & Flexible RAID appliances with slots for hot swap SATA disks
- Check out our full line of EtherDrive® Storage and VirtualStorage Appliances and NAS Gateways

Product Excellence Award Winner LinuxWorld Conference & Expo Best Storage Solution

1. Fast 10 Gigabit Ethernet Storage without the TCP/IP overhead!
2. Unlimited expandability, at the lowest possible price point!!
3. You want more storage…you just buy more disks – it's that simple!!!

Visit us at www.coraid.com

CORAID

The Linux Storage People

1.706.548.7200

www.coraid.com

TECH TIPS

Display date and time in the past or future and pop through directories easily.

Show Date or Time, Past or Future

When working on Linux/UNIX platforms, I frequently find it useful to obtain the date and time in the past or future. Whether scheduling jobs, searching for files from a certain date or determining the day on which a certain date falls, countless scenarios need a routine to compute and display the date and time in the past or future. I searched for a suitable program, but ultimately had to write one myself. This program is called showdate. It is written in the C language originally on UNIX and has been ported over to Linux as well. You can download the code from the *LJ* FTP site: **ftp.linuxjournal.com/pub/lj/listings/issue163/9877.tgz**.

After obtaining the source code, assemble the showdate executable using an ANSI C compiler (either cc or gcc) as shown:

```
# cc showdate.c -o showdate
```

Store the executable in /usr/local/bin or a directory of your choice, and invoke it without any arguments to obtain usage:

```
# showdate
usage: showdate
         [-y [+|-]years]
         [-m [+|-]months]
         [-d [+|-]days]
         [-h [+|-]hours]
         [-M [+|-]minutes]
         [-s [+|-]seconds]
         [-e | -f format]
```

showdate recognizes the following command-line options and arguments:

- -y [+|-]years: Number of years in the past (-) or future (+) offset from the current year.
- -m [+|-]months: Number of months in the past (-) or future (+) offset from the current month.
- -d [+|-]days: Number of days in the past (-) or future (+) offset from the current day.
- -h [+|-]hours: Number of hours in the past (-) or future (+) offset from the current hour.
- -M [+|-]minutes: Number of minutes in the past (-) or future (+) offset from the current minute.
- -s [+|-]seconds: Number of seconds in the past (-) or future (+) offset from the current second.
- -e: Display the time elapsed in seconds since the UNIX epoch (January 1, 1970 UTC).
- -f format: Display the date and time according to formatting directives specified in format.

Options e and f are incompatible. Specifying them together on the command line terminates the program abnormally. The default output of showdate can be tweaked with the formatting directives and argument to -f, which are identical to the ones used by the standard date command. The important thing is that all computations are performed by taking either a positive (future) or negative (past) offset from the current date and time (now), which is its datum.

A good way to become familiar with any tool quickly is to understand how it is used. For example, the command to display the date and time ten years ago, relative to now, would be (output of showdate has been omitted as the results depend on the value of now):

```
# showdate -y -10
```

To find out the epoch seconds elapsed for the above scenario, use:

```
# showdate -y -10 -e
```

A futuristic date of five years, two months and 23 days from now in the YY-MM-DD format would be output as shown below. The plus sign is optional for future dates and the two forms of the command line below are equivalent (the minus sign is mandatory for past dates):

```
# showdate -y +5 -m +2 -d +23 -f %Y-%m-%d

# showdate -y 5 -m 2 -d 23 -f %Y-%m-%d
```

The options can appear in any order, as long as their contextual usage is unambiguous; therefore, the command line above could be written as:

```
# showdate -m 2 -f %Y-%m-%d -d 23 -y 5
```

The +- offsets can be combined in a single command line; however, mixing them up can lead to unexpected and erroneous results. If now is January 1st 2003 12:00:00 AM UTC, showdate outputs:

```
# showdate -m -8 -M 32
Wed May  1 00:32:00 2002
```

The above command displays the date and time in the past—eight months ago but 32 minutes from now, while the one below displays the date and time in the future—8 months from now but 32 minutes ago:

```
# showdate -m 8 -m -32
Sun Aug 31 23:28:00 2003
```

The capabilities of showdate can be leveraged to specify sub-minute job scheduling times. A quick way to schedule a batch job 12 minutes and 35 seconds from now would be:

```
# showdate -M 12 -s 35 -f %Y%m%d%H%M.%S | xargs at -f job-file -t
```

The current date and time is tracked as the number of seconds that have elapsed since the epoch. This number is stored in a signed long, which means that on a 32-bit system, the timekeeping will break on Tuesday January 19, 2038 at 03:14:08 UTC, when the value overflows and becomes negative. An error is returned if the desired date and time exceeds this limit as shown here:

```
# showdate -y 1000
showdate: required time exceeds system limit
```

The presence of whitespace characters in the formatting directive needs to be escaped or enclosed in quotes (single/double). So, the command to display the date and time 18 hours, 30 minutes ago in Year-Month-Day Hour:Minute:Second format would be:

```
# showdate -h -18 -M -30 -f "%Y-%m-%d  %H:%M:%S"
```

showdate cannot obtain the date and time by specifying a weekly offset and by taking a positive or negative offset from any datum, not just the current date and time. Even though showdate has been tried and tested rigorously, it is not perfect. And, if anyone encounters a bug or feels that redesigning the algorithm, implementing coding shortcuts or efficiently using system resources can improve the program, please contact me by e-mail at ssahore@yahoo.com.

showdate was designed for computing and displaying the date and time in the past or future depending on the command-line options, specified as an offset from the current date and time. The next step would be to augment showdate to specify weeks and the ability to change its datum.

—Sandeep Sahore

Push and Pop dirs

The dirs command, combined with pushd and popd, is very effective for tracking users' directory changes. Suppose you have to make some changes to the files present in the following directories:

- /home/sangeeth/soft/release2/src/
- /home/sangeeth/soft/release2/src/show/
- /home/sangeeth/soft/release2/doc/

Instead of noting down the directories on paper, do the following:

```
$ pushd /home/sangeeth/soft/release2/src/
$ pushd /home/sangeeth/soft/release2/src/show/
$ pushd /home/sangeeth/soft/release2/doc/
```

To list all the directories, do the following:

```
$ dirs -l -p -d
```

Suppose you make all the required changes in the first directory (/home/sangeeth/soft/release2/src/). To remove that directory entry from the list of directories that you created earlier, do the following:

```
$ popd
```

The above command removes the topmost directory entry (/home/sangeeth/soft/release2/src/) and performs a cd to the new top directory, which in my case, will be the second directory (/home/sangeeth/soft/release2/src/show/).

Alternatively, one can pop a particular directory from the list of directories by giving the directory ID (the ID is displayed beside a directory when using `dirs -l -p -d`) to the popd command:

```
$ popd +1
```

More options available for using the above commands can be found by viewing the man pages.

—Sangeeth Keeriyadath

Pure64

Power, speed, and performance in one

www.infi-tech.com/pure64

Experince pure 64-bit computing.
Workstation with pure performance, pure speed.

INFITECH

1-800-560-6550

For maximum speed and performance, we recommend PureOS

Thinking about APIs

The historical evolution of Web sites into applications.

REUVEN M. LERNER

People are surprising and unpredictable. In the computer industry, you hear this in nearly every interview with the designer of a popular software package. For example, Perl originally was designed for network programming and text processing. This made it an obvious choice for Web programming, but Larry Wall certainly didn't know or expect that when he first wrote the language, years before the Web was invented.

Users of a software package almost always will push its limits. That's why some of the most successful and popular programs are those that encourage users to go beyond the author's imagination. In the early days of the software industry, such add-ons and plugins didn't exist, which meant that the only way to get new functionality was to lobby the authors and then hope the next release would include the needed features. In the world of open-source software, anyone is theoretically able to add new features, but between the generally small number of core developers and the famously loud debates that sometimes erupt, it can take surprisingly long to add a new feature. (And although it is always possible to fork a project, this has social and technical drawbacks that often outweigh the benefits.)

Some programs have a long tradition of encouraging add-ons. GNU Emacs is best known as a text editor, but it comes with a full-fledged version of the Lisp programming language. You can create just about anything you want in Emacs, and people have done so, including mail-reading programs, Web browsers and an extremely sophisticated calendar/diary. Photoshop caught on among graphic designers not only because of its powerful image editing features, but also because of the large number of plugins that were developed (and sold) for the platform. Microsoft Office, much as it might be reviled by many Linux and open-source advocates, became popular because of its built-in programming language (VBA), as much as for its built-in features. And, of course, the Firefox browser wouldn't be half as useful to me if it weren't for the half-dozen plugins that I have added to my software.

So, users push software to the limits, and software publishers have been able to make their offerings more useful by making it possible to extend their programs. How does this translate into an era of Web-based software? And, what does this mean to us as Web developers?

The answer is the increasingly ubiquitous application programming interface, or API. If you want your Web site to be taken seriously as a platform, and not just an application, you need to offer an API that lets users create, modify and extend your application. APIs are becoming an increasingly standard part of Web-based applications, but despite everyone's use of the term API, that acronym means different things to different people.

Starting next month, I'm going to look at the latest batch of APIs that sites such as Facebook are offering. But this month, I want to take a step back and consider the different types of APIs that software publishers offer. This is useful if you intend to extend, use and work with those APIs. Web development is increasingly a matter of tying together existing functionality from other sites, and understanding these APIs can be quite useful.

It's also important for Web developers to understand the nature of APIs. If you want to create the next Facebook or Google, you're going to need to create more than a winning product. You're going to need an ecosystem of developers and third-party software around your core product. One of the best ways to do this is to create and promote APIs, letting people use your application as a platform, rather than a standalone program. By looking around and seeing what others have done, we can get a better sense of just what the possibilities are and how we might use them.

Read-Only Protocols

In the beginning, when Tim Berners-Lee invented the Web, he imagined it as a read-write medium. But for most people who used the Web during the first decade, it was a read-only medium. You could view Web sites with your browser, fill out forms with your browser, and that was about it. There was no API for reading Web sites; if you wanted to read the content of a site programmatically—for example, in order to create a searchable index of all Web content—you needed to create your own "spider" program, as well as teach it to take apart the HTML.

This changed in the late 1990s, when a number of developers (most prominently, but not exclusively, including Dave Winer) created RSS, standing either for really simple syndication or RDF site summary. In either case, the idea was to create a machine-readable, frequently updated summary of a site's contents. By checking a site's RSS feed, you could learn whether there had been any updates. More significant, RSS feeds were formatted in a standard way, with standard tags, making it fairly easy for programs to poll a feed and extract only the information they wanted.

Unfortunately, the term RSS became both the generic term for syndication and the name for several incompatible (but similar) formats for syndication. A separate group of developers created the Atom protocol, which many people believe is superior to all of the various RSS formats.

RSS and Atom are still popular today. The most common use of these syndication feeds is for blog and news updates, allowing users to keep track of which sites have updated their content. But, RSS and Atom can be used in other ways as well, providing a simple, reliable and machine-readable version of various types of data from a Web site. If you are looking to broadcast regularly updated data, RSS and Atom probably are going to be a good starting point.

For example, the well-known development company 37signals provides an Atom feed of recent activity in its Highrise contact management system. As helpful as it might be to look at your own feed, it would be even more helpful to aggregate multiple people's feeds into a single viewer, allowing, for example, a manager to get a sense of what (and how much) employees are getting done each day.

Read-Write Protocols

The idea that a Web site could provide a regularly updated, machine-parseable version of its content whetted the appetite of many developers for more. Many developers wanted a method to add and modify data, as well as retrieve it.

This came in several different forms, all of which still are used today. The first was XML-RPC, a simple RPC protocol that used HTTP to send an XML-encoded function invocation on a remote server. The server turned the XML-RPC request into a local function call and sent the result of that call (or an error message) in an XML-encoded response. The good news was (and is) that XML-RPC is simple to understand and use, that there are implementations in many different languages, and that they are generally compatible and interoperable.

At the same time, XML-RPC was unable to handle some of the more complex data types that people wanted to use. Plus, it didn't have the official seal of approval or complete standard that would have been necessary for it to enter the corporate arena. So, some of the original XML-RPC developers created SOAP (originally known as the Simple Object Access Protocol, but now an acronym that doesn't expand). SOAP is more sophisticated and complete than XML-RPC, but it had a great many issues with compatibility, implementation and complexity. Today, there are SOAP implementations for many languages, and it continues to be used in a variety of ways, despite some

compatibility issues.

But, at the same time that XML-RPC and SOAP were being touted as the foundations for a new type of interactive, machine-parseable Web, along came Roy Fielding, who described the current state of affairs as unnecessarily complex. He proposed that instead of using XML in both the request and the response, we instead use Representational State Transfer (REST). In other words, the URL should contain everything needed to describe the request, without any additional XML markup or payload. The response, of course, could be in XML or any other format.

The idea of published Web services, procedures invoked via HTTP and URLs that transferred data in XML and other formats, soon became widespread. Creating and using Web services became the biggest thing, with every company talking about how it would take advantage of such Web services. Many standards were proposed for describing and finding Web services; for all I know, these standards still exist, but for the average programmer, they don't, and I'm not sure if and when they ever will.

Read-Write APIs

Given two read-only protocols and three read-write protocols, it was a matter of time before people started to create applications that would take advantage of these. Amazon was one of the first companies to do so, opening up its catalog in a set of Web services now known as Amazon E-Commerce Services, or ECS. Amazon made its entire catalog available via ECS, and it allowed programmers to choose between SOAP and REST. Over the years, ECS has become an increasingly sophisticated and capable system, making it possible to retrieve particular slices of Amazon's catalog and pricing information.

But, retrieving information from Amazon is only half the story: Amazon also makes it possible to manage a shopping cart via ECS and even has some facilities for managing third-party products for sale. Amazon has made a huge commitment to ECS, and a large community of developers and third-party software vendors now exist around this facility. By turning Amazon into a platform for software development, rather than a simple on-line store, Amazon simultaneously has made a community of people dependent on ECS and has created opportunities for the creation of software applications that otherwise never would have been built.

eBay, Google and Yahoo! (among others) also have provided a number of APIs via Web services, which developers can use and access using various protocols. I've read reports, which I can't confirm but that I'm willing to believe, claiming the majority of requests submitted to eBay's servers are through its Web services. Given that most eBay users are not technically sophisticated enough to create their own HTTP clients, we may assume there are a number of software development shops that see eBay as a distribution platform, much as others might see Windows or Linux as their platform.

Google also has exposed a number of its applications to read-write APIs. Rather than use one of the existing protocols, Google uses a version of Atom for both requests and responses, along with a data format it calls GData. There are read-write APIs for a number of Google's applications, including the calendar, Blogger and the spreadsheet program. Programmers no longer are limited by the interface that Google provides to their spreadsheet data; they may create their own programs that use the spreadsheet for storage and retrieval. (One slightly far-fetched example would be the creation of a distributed database server that depended on Google's spreadsheet for locking and version control.)

Although new APIs of this sort constantly are being rolled out, the trend has seemed clear. Make the data easily available and downloadable by the users, in a variety of formats. And, make it possible for them to interact with your Web-based application either using your Web site or (alternatively) the command line or their own home-grown application.

Facebook

Facebook, the social networking site started by Mark Zuckerberg, has become an extremely popular application on the Web. Facebook users can connect with friends, join groups of like-minded individuals and send messages to others. Early in 2007, Facebook became popular among developers, as well as users, for creating a developer API that goes far beyond the APIs I have described above. In a nutshell, Facebook invited developers to create and deploy new applications that are seamlessly integrated into the full Facebook experience.

Facebook isn't the only site that lets you incorporate your own code into the site. However, the style and methods of this integration are deeper on Facebook than I have seen elsewhere. In the Facebook model, your Web application still resides on your server, but its output is displayed inside the user's Facebook page, alongside other Facebook applications. This definitely is something new and exciting; I can't think of any other Web sites that make it possible for an independent developer to distribute code that integrates into the Web site. The fact that you can use whatever language and platform you prefer, communicating with Facebook in a certain way, marks the beginning of a new kind of API, one in which users can affect the Web service as seen by all users, not just one particular user. The only other site I can think of in this camp is Ning, Marc Andreessen's build-your-own-social-network site.

Moreover, Facebook has taken a page from Amazon and eBay, telling developers that they can go wild, using the Facebook network for commercial as well as nonprofit reasons. Google has had a long-standing policy of allowing access to its maps, for example, but only for publicly accessible Web sites and reasons. It remains to be seen whether Facebook's API will continue to be free of charge and open to all.

Something this sophisticated cannot use any one of the protocols that I mentioned above. Rather, Facebook uses a combination of protocols and techniques to communicate with your Web application, making it possible for your programs to display their output alongside other Facebook applications. Moreover, Facebook makes it possible for your application to grab certain pieces of the user's Facebook data, so even though your application doesn't have access to the back-end Facebook database, it still can know (and display) something about the user's friends. Your application even can send messages and notifications to the user's friends, although Facebook has discovered that this can lead to spamming, so it remains to be seen exactly what happens on this front.

Conclusion

Web sites used to be nothing more than an electronic method for publishing and reading basic information encoded in HTML. But, Web sites evolved into applications, which spawned the first generation of APIs that made it possible to read and write your data. Facebook is the first of the new generation of Web sites that look at themselves as a platform more than an application.

And, although Amazon, Google and eBay have demonstrated the importance and potential of a platform-centric view, Facebook is pioneering the incorporation of third-party applications. True, most Facebook applications created to date are simple or trivial. But, we can expect that these applications will become increasingly sophisticated and useful over time. Facebook's willingness to open up to third-party developers is good for everyone—except for competing sites, such as MySpace and LinkedIn, which still appear to see themselves as standalone sites, rather than platforms for new applications.

This month, I explained why I find Facebook's API to be new and exciting. Next month, we'll look at how you can create your own Facebook applications. Even if you aren't interested in creating applications for Facebook, you owe it to yourself to see how the latest generation of Web applications allow themselves to be modified, not just queried.■

Reuven M. Lerner, a longtime Web/database developer and consultant, is a PhD candidate in learning sciences at Northwestern University, studying on-line learning communities. He recently returned (with his wife and three children) to their home in Modi'in, Israel, after four years in the Chicago area.

MARCEL GAGNÉ

Because Nothing Says High Performance Like a Good Race

Got Linux? Got a good accelerated video card as well? Then, get behind the wheel and drive!

Repeat what you just told me, François. No! Don't tell me that! Yes, François, I know I told you to repeat what you told me, but I was sincerely hoping I was mistaken and that I had heard something entirely different. Something like, "Patron, I lost the key to the filing cabinet", not, "I lost the key to the wine cellar." In my first restaurant job, François, this would have been considered a firing offense. No, *mon ami*, I am not going to fire you, but this is terrible! What shall we serve our guests?

Non! I see them walking toward the restaurant now. Quickly, François, think of something. Why you? Because you locked up the wine cellar minutes before opening time, then proceeded to lose the key. That's why. *Quoi?* Serve beer from my personal *réfrigérateur de bière*? That is not a bad idea at all, *mon ami*. You live to serve another day, *non?* Don't look so hurt, François. We have guests. Quickly!

Ah, welcome, everyone, to *Chez Marcel*, where fine Linux fare is served with the most excellent wines from one of the world's premier wine cellars—that is, except tonight. It seems my faithful waiter has lost the keys to the wine cellar. Never fear, *mes amis*, though I love that wondrous liquid fruit of the grape, your Chef also enjoys a good beer, be it lager or ale. So, tonight's menu will be accompanied by selections from my own *réfrigérateur de bière*. Please, sit and make yourselves comfortable. François can offer you several excellent selections, all favorites of mine, including Alexander Keiths' India Pale Ale, Kilkenny Irish Cream Ale, Sleeman's Original, Unibroue Maudite and several others.

As you know, *mes amis*, this issue's theme is high performance, which we all know can refer only to racing and automobiles. If the words high performance and Linux don't immediately generate the same association in your mind, you should know that in point of fact, Linux and car racing go very well together. The 2007 Indianapolis 500 featured the first ever Linux-sponsored car. The brainchild of Ken Starks, aka helios, the Tux500 Project succeeded in its aim to raise enough money to place Tux, the familiar Linux mascot created by Larry Ewing, on the hood of an Indy car. For more details, go to **www.tux500.com**.

Figure 1. The Tux500 Indianapolis racecar—notice the logo on the hood.

In honor of this momentous event, the first race game on tonight's menu, *SuperTuxKart*, features the very same Tux at the wheel. *SuperTuxKart*, maintained by Joerg Henrichs, is an updated and enhanced version of the original *TuxKart*, created by Steve Baker. If you've played the original, you'll be impressed by the new, hugely improved, *SuperTuxKart*. The game features slick new graphics, new tracks, cool characters, racing adventures and more. You take the controls as one of several characters, including Tux, Penny (his friend), Mr. Ice Block, Sushi the Octopus (Figure 2), Yeti and others. You

Figure 2. *SuperTuxKart* features some colorful characters, many with equally colorful names.

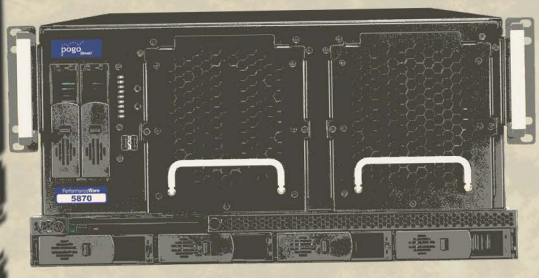
HARD CORE / QUAD CORE

THE QUAD CORE PROCESSORS YOU'VE BEEN LONGING TO SEE

Up to 16 cores and
128GB DDR2 RAM

PERFORMANCEWARE 1680

Up to 32 cores and
256GB DDR2 RAM



PERFORMANCEWARE 5870

SHOCKING PERFORMANCE & RELIABILITY
IN A JUNGLE OF MEDIOCRITY

FIND OUT HOW MUCH HARD CORE COMPUTING POWER CAN BE WEDGED INTO COMPACT SERVERS

Pogo Linux raises the bar again with the all-new 4-way PerformanceWare 1680 and 8-way PerformanceWare 5870 with the new Quad-Core AMD Opteron™ processors. Quad-Core AMD Opteron™ processors feature native quad-core technology on a single piece of silicon so data can move between cores at full speed for more efficient data sharing. AMD's revolutionary Direct Connect Architecture eliminates the bottlenecks inherent in traditional front-side bus architectures, for high-throughput responsiveness and scalability to improve overall system efficiency. Economical in size and price and built for the most demanding database applications, high-performance computing, and server virtualization, the PerformanceWare 1680 and PerformanceWare 5870 deliver. Call Pogo today!

pogo [linux]™

Experience, Imagination, and Support.
Pogo Linux is Hardware Solutions Built for Linux, Built for You.

www.pogolinux.com

To get started, contact us at 888.828.POGO or contact sales@pogolinux

Pogo Linux, Inc. 701 Fifth Ave. Suite 6850, Seattle, WA 98104

AMD Opteron™ 64

Figure 3. Explosions? They never told me there would be explosions!

Figure 4. On the Track with *VDrift* against a Computer Player

can get *SuperTuxKart* from **supertuxkart.berlios.de**.

Your next step is to choose whether you want to race on your own (for practice) or enter a race with other players. These can be other humans (masquerading as *SuperTuxKart* characters, of course), or they can be computer AI players. Once your cast is assembled, it's time to select a race type and course. You can indulge in a Grand Prix event that takes you through all 13 courses, a handful, or simply go for a single race and select a time trial.

Once you are lined up at the starting line, don't be too hasty putting the pedal to the metal or you'll be fined penalty points. Once you are off, your car can do some interesting things besides drive. For starters, you can momentarily jump above your competition and any obstacles in your path. Keep moving and collect spinning cubes along the way, as these contain things like rockets, which you then can fire at opponents and obstacles alike (Figure 3). Try to pick up the spinning fish coins as well—a penguin's gotta eat.

The courses in *SuperTuxKart* are fun, colorful and imaginative. Race though an undersea lair, through the shifting sands of the Egyptian desert or around obstacles in Oliver's Math Glass. And, don't worry too much about falling off the edge of the world or an altogether psychedelic cliff. Your jumping/flying car will carry you back to safety using its combination magnetic levitation system and butterfly wings.

The next item on tonight's menu is a great racing game called *VDrift*. Created by Joe Venzon, *VDrift* is an exciting game based on drift racing. The V in *VDrift* refers to the Vamos Automotive Simulator, written by Sam Varner. Vamos is the physics engine that powers the game. Aside from being exciting and great fun to play, *VDrift* comes with 19 world-class tracks and 28 different car models. *VDrift* is available at **vdrift.net**.

In case you don't know what drift racing is (your humble Chef did not), it's a form of racing where the driver employs "drifting" to control the car. Doesn't help? Well, this quote from Wikipedia might clear things up: "A car is said to be drifting when the rear slip angle is greater than the front slip angle and the front wheels are pointing in the opposite direction of the turn." In other words, the car is turning one way, but the wheels are pointing in another direction. This, strange as it may seem, is not an accident. Remember, the driver is in control. As the driver, what you are doing is sliding into a turn to avoid slowing down as much as possible.

When the game starts, you can select the car you want and, in some cases, the color. Where you race is another option. Choose a racetrack or go for a relaxing, high-speed

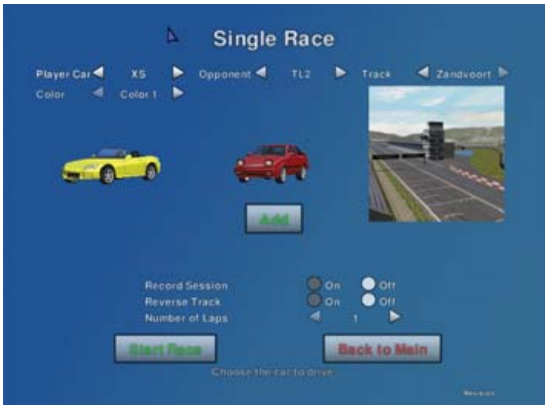


Figure 5. *VDrift* lets you select from several different cars, colors and racing venues.

Note:

All these packages are available as source code bundles from their respective Web sites. Before you start compiling (unless you want to, of course), check your distribution's software repositories for binary packages. In all cases, I had no trouble finding packages.

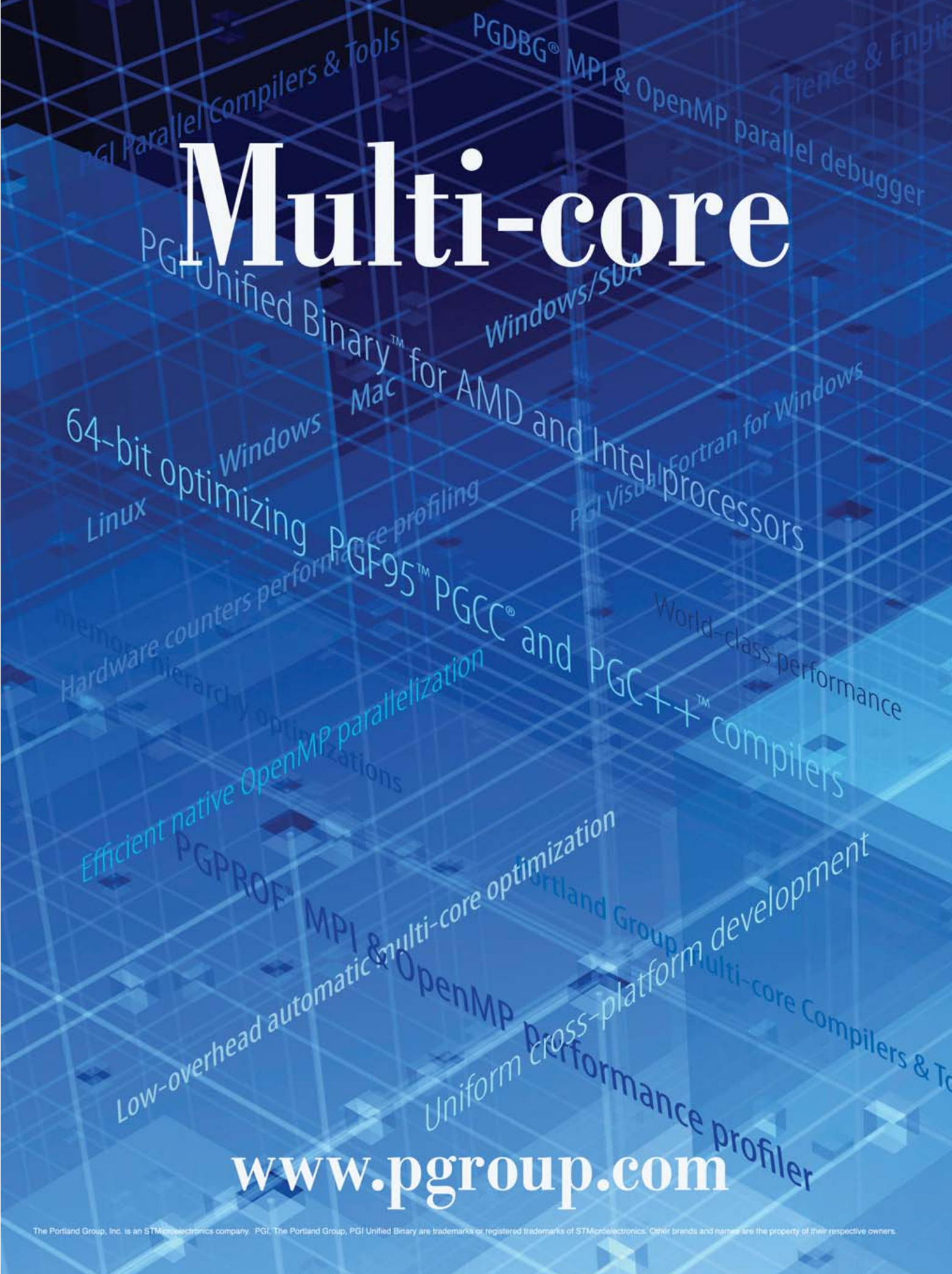


Figure 6. Yes, it is exactly what it looks like. There is a huge gap in the road. Hammer down, jump, keep it straight and keep your cool.

drive in the country. Practice races put you on the track with no other cars to worry about. Races pit you against an opponent. On a single computer, you can play *VDrift* alone or against a computer opponent, at which point, you also can decide on your opponent's car model and color (Figure 5). *VDrift* also can be run as a server, at which point, your friends can join in for a networked game.

Several option controls exist to make the game more realistic, and more difficult, if you so choose. For instance, you can decide what effect speed has on steering by adjusting a slider to a desired percentage. The higher the percentage, the more difficult it is at high speeds. You also can choose to map different keys for the gas pedal, brakes and steering. Shifting and clutch controls can be defined as well. I confess that I found it much easier to set up to use an automatic transmission on my first few races. That brings up another point—make sure you are in gear before you start. It took me a few tries before I realized that I was going backward. Press the S key to start, then press 1 to get in gear. If you have selected automatic, shifting will happen automatically for you after this.

There are more options and settings, which I'll let you discover, but I do want to mention one other group of settings that may be important. Given that this an OpenGL game and that it does require hardware 3-D acceleration, owners of somewhat less-powerful video cards may find the game more responsive by toning down some of the effects.

As you race, you can change your viewing angle by pressing F1 through F5. If you find yourself totally taken with the action on-screen and you feel the desire to preserve the moment, you can get a screenshot of the action at any time by pressing F12. Those images will appear in the folder .vdrift/screenshots in your home directory.

The final item on tonight's menu is the perfect selection for those of you who experience a kind of mania when it comes to racecar driving. The game, aptly named *ManiaDrive*, is an incredibly fast-paced game with rapid turns, nerve-wracking jumps (Figure 6) and a driving, rocking, soundtrack (*ManiaDrive* is actually a clone of Nadéo Studio's *Trackmania* game). *ManiaDrive* features a training mode designed to prepare you for the real thing and a set of complex tracks that can be played locally or on-line with other players. You can get *ManiaDrive* from **maniadrive.raydium.org**.

When you start *ManiaDrive*, make sure you go through the Beginner story, which will guide you through the various moves that are expected of you. When you are ready for the big time, choose the Pro story mode. This game is more an endurance and skill test than a race. Sure, there's a clock ticking and your speed is tracked (on-line, against others, no less), but races last only a short time, some less than a minute. If you are the type that gets bored quickly, *ManiaDrive* is for you.

Speaking of fast, is it possible that the time has gone by so quickly? *Mon Dieu!* And yet, it does appear that closing time is almost upon us. Well, *mes amis*, despite the apparent crisis that we apparently faced tonight, I dare say the beer selection was most refreshing. I can tell from your approving nods that you agree with me. Nevertheless, tomorrow, I shall have the wine cellar lock changed, and next time, our award-winning cellar will once again be open. But for now, François will pour you another beer that you may enjoy at your leisure while you enjoy another race. Remember, *mes amis*, that sitting in front of your Linux systems driving a virtual car is the only way to drink and drive safely. Raise your glasses, *mes amis*, and let us all drink to one another's health. *A votre santé! Bon appétit!*■

Marcel Gagné is an award-winning writer living in Waterloo, Ontario. He is the author of the all-new *Moving to Free Software*, his sixth book from Addison-Wesley. He also makes regular television appearances as Call for Help's Linux guy. Marcel is also a pilot, a past Top-40 disc jockey, writes science fiction and fantasy, and folds a mean Origami T-Rex. He can be reached via e-mail at mggagne@salmar.com. You can discover lots of other things (including great Wine links) from his Web site at www.marcelgagne.com.

Resources

ManiaDrive: **maniadrive.raydium.org**

SuperTuxKart: **supertuxkart.berlios.de**

VDrift: **vdrift.net**

Marcel's Web Site: **www.marcelgagne.com**

The WFTL-LUG, Marcel's Online Linux User Group: **www.marcelgagne.com/wftllugform.html**

Polywell's Ultimate Linux Systems

More Choices, Excellent Support Service, Great Value!

1U Value Servers Starts at $399, Up to 8GB RAM

(custom config. available)

$399 1U-485Ax Sempron 3000+, 1GB DDR2, 80GB HD
(For Volume Purchase Only)
$499 1U-485Ax Athlon64 3500+, 2GB DDR2, 80GB HD
(For Volume Purchase Only)
$699 1U-690GA Athlon64 X2 Dual-Core, 4GB DDR2, 2x160GB HD (Dual LAN +$45)
$999 1U-690GA Athlon64 X2 Dual-Core, 8GB DDR2, 2x160GB HD (Dual GigaLAN +$45)

Linux Appliance Starts at $299

Poly 6100Nx Athlon64 X2, 512MB RAM
Nvidia 6100 Graphics, 80G, CD-ROM $299

Poly 7050A Athlon64 X2 4200+, 1GB DDR2
Nvidia 7050 Graphics, 500G, DVD-RW $499

(OEM /ODM Service Available)

1U Advanced Servers, Up to 64GB RAM, 4TB HD

$1,799 1U4S-690A Athlon64 X2 6000+, 8GB DDR2
2TB 4x500GB HD, Dual Gigabit LAN
$2,999 1U-2500A16 2 x Opteron 2212, 16GB ECC DDR2,
1TB 4x250GB HD, Dual Gigabit LAN
$4,999 1U-2500A16 2 x Opteron 2216, 32GB ECC DDR2,
2TB 4x500GB HD, Dual Gigabit LAN (Option: 64GB+4TB HD)
$10,999 1U-8415SS32 4 x Opteron 8212, 64GB ECC DDR2,
2x73GB SAS HD, 3x Gigabit LAN (Option: 128GB RAM)

Low-Cost NAS Storage
2.0TB Starts at $999

$999 Netdisk 4000 2.0TB 4x500G
$1,499 Netdisk 6000A 3TB 6x500G (2U)
$2,999 Netdisk 6000B 6TB 6x1000G (2U)

1U Twin Servers, 1U 8-Way Quad Opteron

$4,999 1U-Twin 2 x 2 Dual-Core Processors, 2 x 8GB ECC DDR2,
2 x Dual 250GB HD, 2 x Dual Gigabit LAN
$5,999 1U-8415A 4 x Opteron 8212 Dual-Core, 16GB ECC DDR2,
1.5TB 3x500GB HD, Dual Gigabit LAN
$7,999 1U-8415SS 4 x Opteron 8212 Dual-Core, 32GB ECC DDR2,
2x73GB 15K RPM SAS HD, 3x Gigabit LAN

High-Density Multi-Processor Servers

1U to 5U, Quad to Eight Processors, Up to 128G RAM

$11,999 2U-8425SS 4 x Opteron 8212, 64GB ECC DDR2,
4TB 8x500GB RAID-5 Storage, 3 x Gigabit LAN
$32,999 2U-8425SS 4 x Opteron 8212, 128GB ECC DDR2,
2x73GB 15K RPM SAS HD, 3x Gigabit LAN
$16,500 5U-8850T5U 8 x Opteron 8212 , 64GB RAM,
2TB 4x500GB HD, 3 x Gigabit LAN
$36,999 5U-8850T5U 8 x Opteron 8212, 128GB RAM,
4TB 8x500GB HD, 3 x Gigabit LAN

6TB 2U Storage Server
2012SC-2055A

4TB 8x500G, Opteron $2,999
6TB 12x500G, Opteron $3,999

Blade Servers - 10 Dual or Quad Processors Blades

$13,999 8U 10 x 2055A Blades
10 x (Dual Opteron 2210, 4GB RAM, 80G HD)
$24,999 8U 10 x 2500A16 Blades
10 x (Dual Opteron 2212 , 16GB RAM, 80G HD)
$46,999 8U 10 x 8450A Blades
10 x (Quad Opteron 8212, 16GB RAM, 80G HD)
$69,999 8U 10 x 8450A Blades
10 x (Quad Opteron 2214, 32GB RAM, 80G HD)

24TB 4U Storage Server
4024SS 4U 24-Bay

12TB 24x500G, Opteron $7,500
24TB 24x1000G, Opteron $12,999

AMD Dual-Core technology enables one platform to meet the needs of multi-tasking and multi-threaded environments; provides platform longevity

Polywell OEM Services, Your Virtual Manufacturer
Prototype Development with Linux/FreeBSD Support
Small Scale to Mass Production Manufacturing
Fulfillment, Shipping and RMA Repairs

AMD 64 Opteron™

AMD 64 Athlon X2

- 20 Years of Customer Satisfaction
- 5-Year Warranty, Industry's Longest
- First Class Customer Service

888.765.9686
www.Polywell.com/us/LJ

Polywell Computers, Inc 1461 San Mateo Ave. South San Francisco, CA 94080 650.583.7222 Fax: 650.583.1974

POLYWELL

DAVE TAYLOR

Keeping Score in *Yahtzee*

Push an array sort out to the sort function instead of writing a sort routine.

Last month, I started talking about how to use some simple shell scripting techniques to create a computer simulation of the popular dice game *Yahtzee*. I'm not going to write the entire game (the computer player would be darn complicated for a shell script, for one thing), but let's spend one more column looking at some of the basics of scoring before we move on to other topics.

One nice thing about *Yahtzee* as a computer game is that there really isn't much work for the computer to do, because the game works fine as a solitaire variation: you simply can play to try to maximize your score and see how you do, competing against your previous high score.

The basic idea of *Yahtzee* is that you roll five dice up to three times to get the best possible set of values. Those are then scored against a fixed score pad that offers specific points for specific combinations, ranging from one point per "one" in your roll up to 50 points for five of a kind (a "Yahtzee", in game parlance).

A quick visit to Google identifies an on-line *Yahtzee* score sheet; take a look at it before we proceed: **www.gamingcorner.nl/images/sheets/yahtzee-3.pdf**.

In many ways, it's five-card draw poker with dice, so the first section of the score sheet is for sets: ones, twos, threes and so on, where your score for each is the sum of that value on the dice. The second section is other poker hands, including three of a kind, four of a kind, small straight (four of the five dice have sequential values), large straight (all five are sequential), a full house and one or more Yahtzee rolls.

You can get bonuses for achieving certain levels, including a very nice 35-point bonus for attaining at least 63 on the top section, but the score sheet itself is straightforward.

The key to the game, then, is to figure out how to score a given roll. If you roll four ones, do you want to score that as your four of a kind or your ones? What if they're both filled in already? Fortunately, we're going to defer to the player for that, but that still leaves us with the data question of how to model the different boxes on the score sheet and the interface question of how to prompt the user to select which box to score. Let's take a look.

Modeling the Score Sheet as an Array

As with most of these types of data structures, we'll use an array to model the score sheet. Count the boxes on the score sheet, and you'll see 13 boxes total, including the special Yahtzee box where you can roll—and get credit for—more than one (so a game typically has 13 rolls, but it could have more).

If we initialize the game by filling in all the array values with a known stop value, say -1, then the test for whether a given box has been filled in is easy:

```
if [ scoresheet[1] != -1 ] ; then
  echo "1: Your one's" ; fi
```

The trick is that we also want to pre-qualify these options. There's no point in prompting players to select their roll as a Yahtzee if they didn't get five of a kind. This proves to be a bit tricky, so as a first step, I'm going to tweak the code to order the dice in the array in ascending order after each roll automatically.

It might be a long trip to accomplish the task, but rather than write a sorting routine in the script, I'm just going to push out the task to the sort function, then read the results back in and fit them into the individual slots of the dice array. Sound complicated? It is, rather:

```
function sortDice()
{
   sorted="$( ( echo ${dice[1]} ; echo ${dice[2]}
     echo ${dice[3]} ; echo ${dice[4]}
     echo ${dice[5]} ) | sort )"

   index=1
   for value in $sorted ; do
     dice[$index]=$value
     index=$(( $index + 1 ))
   done
}
```

You can see here that I'm using a temp variable called sorted to store the resultant values, and that I'm using a subshell—that's the $( ) notation—to do the actual work. The hardest part of this little function is to figure out how to put the values back into the array once everything's sorted properly, and that's accomplished with the for loop.

Notice that, by a lucky coincidence, for loops automatically step through fields separated by white space (spaces and carriage returns), so it's perfect for breaking the resultant sorted sequence back into individual values.

We're running low on space this month, and I'm afraid I've ended up spending quite a bit of time talking, rather than coding. I'll make it up to you, dear reader, next month!■

Dave Taylor is a 26-year veteran of UNIX, creator of The Elm Mail System, and most recently author of both the best-selling *Wicked Cool Shell Scripts* and *Teach Yourself Unix in 24 Hours*, among his 16 technical books. His main Web site is at www.intuitive.com, and he also offers up tech support at AskDaveTaylor.com.

SUPERMICRO®

SuperBlade™

Much Better TCO than 1U Servers

Intel Processor Blade w/ 3.5 in. Hot-swappable Drive Bays

AMD Quad Processor Blade

Chassis Management Module

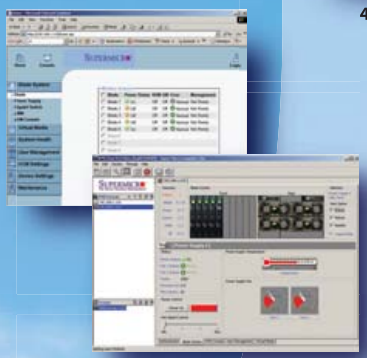
Gigabit Ethernet Module

4x DDR Infiniband switch

160 Cores (40 processors) and 640GB Memory in 7U 90%+ Power Efficiency

- Features Intel® Xeon® and AMD Opteron® DP/MP
- Holds up to 10/14 server blades
- Six enclosures fit a 42U rack
- 90%+ High-Efficiency, 3+1 redundant power supplies
- Chassis management modules
- Gigabit Ethernet switch modules
- Gigabit Ethernet pass-thru modules
- InfiniBand switch modules (20 Gb/s x2)

CMM Web-based GUI



CMM IPMI View

Application-Optimized for:

Enterprises, Financial Services, Databases, Data Centers
Science Labs, HPC, Personal Super Computer

www.supermicro.com

AMAX	Arrow Electronics	ASI	Bell Micro	Ingram Micro	MA LABS	Synnex	Tech Data
1-800-800-6328	1-888-427-2250	1-800-2000-ASI	1-800-232-9920	1-800-456-8000	1-408-941-0808	1-800-756-5974	1-800-237-8931
www.amax.com	www.arrownacp.com	www.asipartner.com	www.bellmicro.com	www.ingrammicro.com	www.malabs.com	www.synnex.com	www.techdata.com

JON "MADDOG" HALL

Navigating by the Sun

Lowest TCO for sale to highest bidder.

"Dead reckoning", cried Davi. "What is that?"

I had been adjusting the compass of my sailboat, the *Agape*, measuring for possible magnetic influences that would affect the readings, and I had mentioned the dead reckoning method of navigation to Davi. Now, I knew I was going to pay the price of explanation.

"Dead reckoning is the method of determining where you are by knowing where you have been, adding to that the direction and speed you have been traveling, and the length of time you have been going in that direction to determine a new position. It was used for many hundreds of years at sea", I told him.

"Wow, that must have been hard", said my young friend. "And it must have generated many errors."

"Yes", I answered, "and the method was blamed for the loss of hundreds of ships and thousands of lives, including the wreck of the *HMS Association* and several other ships from the British fleet off the Isles of Scilly on October 22, 1707—1,400 British sailors died."

I continued, "This accident led John Harrison, a carpenter, to develop the marine chronometer, a clock accurate enough to calculate longitude easily while at sea. Harrison's life is a fascinating story of perseverance and hard work, which led to many important inventions, including bi-metal strips and the roller bearing. Unfortunately, England's Board of Longitude kept rejecting the work done by Harrison, citing inaccuracies in the running of the clocks, and it was felt that part of this rejection was because Harrison was a humble carpenter and not a gentleman. Eventually, the king had to intervene to see that justice was done."

Aboard the *Agape*, e-mail is my prime method of communication, and lately, I have been receiving messages from Sun Microsystems, telling me I should be switching from Linux to Solaris. They cite many "reasons" for doing this, and although some are features of Solaris that have not yet been incorporated into Linux, most of them reek of marketing hype.

The first message I received (May 25, 2007) told me that "Solaris offers one thing Linux does—and 600 things it does not. The Solaris OS is free and open source—same as Linux." Yet, nowhere in the rest of the message does Sun identify those 600 things that Solaris offers. Nor could I find them listed on Sun's Web site where the links guided me. Although the message does identify several features that Solaris has "built in", some of those features are offered by major Linux distributions, and others can be added by "rolling your own" in the Linux space.

Two of the major features Sun offers are "low-cost licensing and support options" and "a disruption-free migration path from the difficulties of a Linux platform to the ease and value of a Solaris 10 solution". Hmm, sounds a bit like marketing hype to me.

Then, on June 27, 2007, I received another letter from Sun:

> A lot of myths have been circulating about Linux—that it's low cost; that it runs more apps; that it's pervasive...but when you look at the facts you see a different story.
>
> Red Hat 5.0 does not save you money, runs fewer certified applications, and makes migration more difficult. These are just three of the many compelling facts that should make you wonder, "why are you running Linux when you could be running Solaris 10 OS?"

Now, Total Cost of Ownership (TCO) studies abound showing how Microsoft is cheaper than Sun, Sun is cheaper than Linux, Linux is cheaper than Microsoft, and so on. Normally, these TCO studies show whatever the sponsoring company wants them to show, and it does not take heaps of intelligence to realize that a TCO study that does not show the desired "winner" would not see the light of day. Although TCO is important (because if you cannot afford the solution, you cannot implement it), the really interesting indicator is Return on Investment (ROI) and long-term benefits, which I call Value of Investment (VOI). VOI is usually what a company wants for the long run.

Regarding the claim that Red Hat 5.0 "does not save you money", Sun offered one example of a solution offered to Marc Andreessen. Sun's solution was cheaper than "Linux on Intel commodity hardware". Okay, fine. But, this particular customer also was Marc Andreessen, and having worked in the UNIX Group at Digital Equipment Corporation (DEC) for 16 years, I can imagine the discussion between the various management groups inside Sun to make sure it won this account and to make sure it was "a good deal", of which there is no detailed analysis of the terms listed in the e-mail or on the site. There is nothing wrong, illegal or even immoral about this deal and its subsequent use in marketing (and certainly not on Marc's part); it's just a bit opportunistic by Sun to use a well-known name on a single deal to make its point—not scientific, by any means.

As for the claim that Solaris supports more "certified applications" than Red Hat v5.0, Sun used the application base of its SPARC and its x86 architectures *combined*. How many people want to run both a SPARC and an Intel box just to get the complete suite of applications? Additionally, the Solaris OS version had been out for a while, giving independent software vendors (ISVs) a chance to certify their applications. Red Hat's v5.0 had been out for only a month or so at the time of the "data effectiveness" cited in an e-mail message on April 4, 2007. This is hardly enough time for the ISVs to certify their software on top of Red

Hat's newest release and and get it into Red Hat's system. Again, there's nothing wrong or illegal, it's just a little sleazy as a marketing ploy—okay, more than a little sleazy.

Sun's claim that Red Hat "makes migration more difficult", brings about the question "migration from what?" Last night, I went to my Linux user group meeting, and the speaker, who was running a mixed Solaris and Linux shop, lamented how hard it was to work on a Solaris system because of the antiquated and nonstandard set of utilities on the upper levels of the operating system. His presentation confirmed some of the issues I knew about and showed that being a Sun customer was not all peaches and cream. But then again, I do not expect Sun's marketing group to haul out the spoiled milk.

Sun also claims binary compatibility between releases, but when you actually read Sun's terms and conditions on binary compatibility, it basically says Sun guarantees your program to run as long as you did not use any interfaces that did not change or were not deleted. This is a guarantee? It sounds like most of the warranties in the EULAs of most software products I have seen—basically worthless.

I received a final e-mail message on June 28, 2007, inviting me to tell Sun why I used Linux, and offering me a free 4GB iPod Nano if I came in for a one-hour meeting to listen to why I should use Solaris. I would get the iPod if I signed up *and* if I was qualified *and* if the number of iPods (25) had not been depleted. Hmm, not that I even want an iPod, but what was the chance of getting that iPod? That's not illegal or immoral or even sleazy—just marketing.

Why am I making a big deal about what obviously is a badly positioned marketing campaign? After all, I was in the marketing group for DEC the last eight years of my employment there. I know what it's like in a large company with many product groups and many marketing strategies that often run in conflict with each other. I imagine some people inside Sun really believe in FOSS and the community, and gnash their teeth whenever they hear of such things. I even know one.

First, Sun purports to be a friend of the Free and Open Source community and a reseller of Linux and Linux hardware. It has had four of its server systems certified by at least one Linux distribution, and it is listed as a Gold Partner of that Linux distribution. I assume that it is making money doing this and perhaps even is selling services for those servers and the Linux systems on them.

But, then I hear reports (and receive e-mail messages) that show this is simply a bait-and-switch program—to get Linux customers onto its customer list and convince them to switch to Solaris. With partners like this, you really don't need competitors.

Second, Sun has had a long and troubled history with maintaining the consistency it brags about—corporate decisions that went in the face of what its customer base really wanted, such as:

- The *very painful* migration of BSD SunOS to System 5.0-based Solaris, and the needless renaming (and subsequent system administration issues) of the BSD-based SunOS code base to Solaris 1.x (even though it was the same code base).

- Supporting and then decommitting from Solaris Intel, not once, but twice.

- Ignoring the fact that SPARC is big-endian and Intel is little-endian, so that binary data often is incompatible. ("NFS takes care of that", says Sun at a tradeshow. "No, it does not", says maddog back to them at the same show.) Endianism was a point that Sun kept pushing as long

as Big-Endian machines were prominent in the marketplace, but it has ignored this ever since Intel/AMD reached dominance and SPARCs were the poor kids on the block.

Third, Sun needs to decide what it wants to be when it grows up. It continues to spend millions of dollars to duplicate a base operating system, when it could be addressing customers' needs by better supporting the Linux community and recognizing that the real enemy is a certain closed-source, proprietary operating system from a company that does not allow real competition. Sun needs to redefine its strategy and become a solutions company (like IBM and HP are doing), while not abandoning its line of VARs and resellers that made Sun great. Sun needs to learn how to compete on the really equal playing field that computer products are now proceeding toward. It can do that, but only if it is smarter about it.

Fourth, Sun needs to clean up its marketing messages. It claims to have been "founded as an open-source company". In reality, Sun was one of the companies that took UNIX into a binary distribution model. Although it is true that this model was somewhat forced by AT&T's arcane licensing strategies of the time, it was this movement of UNIX companies to restrict the source code distribution that helped motivate Richard Stallman to produce the GNU suite and the University of California Berkeley to complete the first freely distributable Berkeley Software Distribution.

I do not want to imply that Sun has never helped the Open Source community or contributed to its well being. Certainly, through time there have been stellar examples of individuals at Sun that have contributed more than their share to the community overall. But, let's put it into the perspective of a company that has had its share of proprietary, closed-source products and standards wars.

Sun does have some good products. Its work in multicore chips and multicore development tools (illustrated in an article this month by Roman Shaposhnik, a Sun employee, on page 80) is a good example of Sun leveraging off its technologies. Sun also has been doing some good work in environmentally sound server systems and solutions.

Unfortunately, there is a big difference between having good people working for you who believe in free and open-source software and what the corporate, management and marketing culture states is the direction of the company.

Here is my beef: I am a Linux Guy. I believe in free and open-source software, and this means more than just the open-source licenses that Sun offers. It means that customers easily can contribute back to the path that Sun as a company takes, and it means that customers easily can contribute back to the code that Sun writes. It means that as a business partner, I can feel free to propose my solutions to my customers without having my partner run roughshod over me. I have found my solution to free and open-source software in the Linux kernel and the people and companies building and contributing to it.

That said, when I engage customers, I engage them with the concept of solving their problems, and I am not above using a less-open solution than what I might otherwise find if I think it is worthwhile for the customer's VOI. I view this as a tactical solution, although the strategic (and long-term) solution may still be FOSS. This even might lead to me advocating a Solaris solution for my customer. But this is what I, as the solution provider, work out with *my* customer.

On the opposite side, being a Linux Guy also means that when I engage Sun as a supplier of Linux servers for my customers, I want to feel safe that I can turn my back for a moment and not have the Sun people who service or resell those servers approach my customers and tell them that my solution is bad, particularly when that story is based on half-baked marketing messages with no staying power.

When I was at DEC, I was making fairly good money in a fairly stable job with good benefits. I saw no issue in selling both Digital UNIX and Linux on the Alpha. What my customers asked for, I sold them. When customers asked my opinion, I gave them the best answer I could for their solution. At that time, Digital UNIX was a commercially robust and feature-full implementation, and Linux lacked in a lot of the features that Digital UNIX offered. I did not tell customers to buy Linux or buy Digital UNIX, because they both made money for Digital. I could, however, see the writing on the wall. I knew that over time Linux would have all the features of a fully robust commercial system, that it would be used as the research arm for computer science, and that companies pouring $200–300 million a year into developing their "own version of UNIX" would not be able to compete with an operating system and kernel being developed jointly by many companies and many minds.

I left that stable job at DEC, because I believed in the Linux community and the free software model, and because I did not want to embarrass DEC by actively and aggressively criticizing its biggest business partner, Microsoft.

Now, Sun runs the risk of alienating what could be its biggest and best business partner, the Linux community, including Linux VARs and Linux resellers.

Just as ships no longer use marine chronometers and sextants to "follow the Sun", but use GPS systems instead, companies no longer guide their businesses with half-baked marketing statements.

Sun's marketing group needs to understand that this is no longer the time of sailing ships and dead reckoning. You cannot just start from where you have been and hope to end up where you want by setting some course and sailing full steam ahead. You will find yourself on the rocks.

The product era of computer science is sinking, and now the era of service-oriented computing is well underway. Sun's marketing and business practices need to move into the 21st century, and Sun needs a new vision for how it is going to provide solutions for its customers and partners and move the company forward. I really hope Sun finds it.■

Jon "maddog" Hall is the Executive Director of Linux International (www.li.org), a nonprofit association of end users who wish to support and promote the Linux operating system. During his career in commercial computing, which started in 1969, Mr Hall has been a programmer, systems designer, systems administrator, product manager, technical marketing manager and educator. He has worked for such companies as Western Electric Corporation, Aetna Life and Casualty, Bell Laboratories, Digital Equipment Corporation, VA Linux Systems and SGI. He is now an independent consultant in Free and Open Source Software (FOSS) Business and Technical issues.

DOC SEARLS

The Usefulness Paradigm

If you don't like the usefulness paradigm, submit a patch.

I don't write code, but I do write prose. It's different, but there are similarities and overlaps. Code writers have also supplied me with a useful cache of metaphors. For example, I can explain my writing method as a debugging and patching process—and not just because those are metaphors for editing. Rather, it's because I regard most of my writing less as finished work than as useful information that is always subject to improvement. I'm not just here to express. I'm here to help. (Okay, sometimes I just joke around, but that's different.)

My model is the most useful tome I've ever read, *The Elements of Style*, by William Strunk Jr. and E. B. White. Better known just as "Strunk and White", the book's 1.0 version (**en.wikisource.org/wiki/The_Elements_of_Style**) was written in 1918 by Strunk alone, when he was a professor of English at Cornell University. It was privately published and only 43 pages long. It began:

> This book is intended for use in English courses in which the practice of composition is combined with the study of literature. It aims to give in brief space the principal requirements of plain English style. It aims to lighten the task of instructor and student by concentrating attention (in Chapters II and III) on a few essentials, the rules of usage and principles of composition most commonly violated. The numbers of the sections may be used as references in correcting manuscripts.

The book is a model of brevity and pith. The paragraph above is first of the five that make up Chapter I, which fits on a single page. Here is Strunk's outline for Chapters II and III:

> **II. Elementary Rules of Usage**
>
> 1. Form the possessive singular of nouns with *'s*.
>
> 2. In a series of three or more terms with a single conjunction, use a comma after each term except the last.
>
> 3. Enclose parenthetic expressions between commas.
>
> 4. Place a comma before *and* or *but* introducing an independent clause.
>
> 5. Do not join independent clauses by a comma.
>
> 6. Do not break sentences in two.
>
> 7. A participial phrase at the beginning of a sentence must refer to the grammatical subject.
>
> 8. Divide words at line-ends, in accordance with their formation and pronunciation.
>
> **III. Elementary Principles of Composition**
>
> 9. Make the paragraph the unit of composition: one paragraph to each topic.
>
> 10. As a rule, begin each paragraph with a topic sentence; end it in conformity with the beginning.
>
> 11. Use the active voice.
>
> 12. Put statements in positive form.
>
> 13. Omit needless words.
>
> 14. Avoid a succession of loose sentences.
>
> 15. Express co-ordinate ideas in similar form.
>
> 16. Keep related words together.
>
> 17. In summaries, keep to one tense.
>
> 18. Place the emphatic words of a sentence at the end.

Just reading that outline makes you a better writer. Digging deeper has the same effect. Take #13 from Chapter III: "Omit needless words." It begins:

> Vigorous writing is concise. A sentence should contain no unnecessary words, a paragraph no unnecessary sentences, for the same reason that a drawing should have no unnecessary lines and a machine no unnecessary parts. This requires not that the writer make all his sentences short, or that he avoid all detail and treat his subjects only in outline, but that every word tell.

Familiar? Seems to me the same principle applies to writing code. One difference is that writers of prose generally need to be instructed on the value of brevity while writers of code do not. That may seem a harsh or inaccurate statement, but I make it because code is purposeful in ways prose does not have to be. In fact, blather is sometimes the very purpose of prose. Garrison Keillor once called English "the preacher's language" because "it allows you to talk until you think of what to say."

I suppose you can blather in code too. Back in the late 1990s, I was sitting in the office of Phil Hughes, *Linux Journal*'s founder and publisher. He had just come from an argumentative session with a group of geeks in another room. Imagining I would know what he was talking about, he threw a pile of code printout onto his desk. "Look at that!", he said. "Six pages of Perl! I could have done the same thing in one page of awk!"

I'm not saying Perl is blather, by the way (though Phil might). I am saying that omitting needless words is a code-writing value for which prose writers require help from the likes of William Strunk.

Version 1.5 of Strunk and White came with a 1935 revision by Edward A. Kenny titled *The Elements and Practice of Composition*. Strunk died in 1946. Far as I know, his book didn't change until 1957, when the writer E. B. White, once a student of Strunk's, came again upon "the little book" and set about improving it with two short additional chapters and a return to the original title. This first edition of what came to be called "Strunk and White" was version 2.0 of what began as Strunk's alone. In the 86-page 1979 edition, White explained:

> A second edition of the book was published in 1972. I have now completed a third revision. Chapter IV has been refurbished with words and expressions of a recent vintage; four rules of usage have been added to Chapter I. Fresh examples have been added to some of the rules and principles, amplification has reared its head in a few places in the text where I felt an assault could successfully be made on the bastions of its brevity, and in general the book has received a thorough overhaul—to

> correct errors, delete bewhiskered entries, and enliven the argument.

White died in 1985. The 1999 fourth edition of the book has a forward by White's stepson, Roger Angell, plus an index, a brief afterward by Charles Osgood and a glossary by Robert DiYanni. The current entry in Wikipedia notes, "An anonymous editor modified the text of this 1999 edition. Among other changes, he or she removed White's spirited defense of 'he' for nouns embracing both genders. See the 'they' entry in Chapter IV and also gender-specific pronouns." In his forward, Angell puts it more gently, "This edition has been modestly updated...with a light redistribution of genders to permit a feminine pronoun or female farmer to take their places among the males who once innocently served him. Sylvia Plath has knocked Keats out of the box, and I notice that 'America' has become 'this country' in a sample text, to forestall a subsequent and possibly demeaning 'she' in the same paragraph."

Astute readers of Strunk, White and *Linux Journal* will observe that we violate one of the former pair's original commands by putting our punctuation outside our closing quotation marks, unless it is part of the quotation. This complies with a *Linux Journal* copy-editing policy established by Phil Hughes in 1994, in compliance with what had already become geek vernacular. One of my first arguments with Phil was over this very practice. I lost.

Today if you look for *The Elements of Style* at Amazon.com, you'll find three different books: The 1999 fourth edition, Strunk's original edition and *The Elements of Style Illustrated*, which (Wikipedia tells us) is the fourth edition plus illustrations by Maira Kalman. Somehow I can't help regarding the three as "distributions" of Strunk's patched kernel, each useful in its own way.

It's also interesting to read and compare the discussion and history pages behind the Wikipedia entry for *The Elements of Style*. The former calls to mind the Linux-Kernel Mailing List (LKML), while the latter serves as a kind of code repository. These analogies are far from precise, but there is a common purpose in usefulness.

What brought me to this fresh appreciation of usefulness was a blog post by Alan Mitchell titled "Beyond the persuasion paradigm" (**rightsideup.blogs.com/my_weblog/2007/08/beyond-the-pers.html**). He writes:

> Our modern commercial setup is organised around a persuasion paradigm, where the driving force in commerce is companies and their attempts to persuade individuals—so-called consumers—to buy their products or services (and not those of their rivals).
>
> We are, however, groping our way towards a personal decision-making paradigm, whose centre of gravity is helping individuals make and implement better decisions at lower cost (where "cost" includes time and hassle cost, as well as money cost). This is being made possible by an information age: our increasing ability to search for, gather, store, filter, edit, slice and dice, analyze, share the information we need to for our decision-making.
>
> Better personal decision-making (and implementation) is an era-defining shift for two simple reasons: it shifts the focus of value creation from companies and their goals to individuals and their goals, and it encompasses all aspects of production, distribution and exchange (and a whole lot more)....

There is in this a declaration of independence for individuals, in heed of the need to equip human beings with better ways to learn, to make choices and to exercise their autonomy. What's not clear, and needs to be, is the role played by programmers in equipping everybody with the freedom to escape the old persuasion paradigm, and a value system in which the dependencies that matter most are ones where buyers rely entirely on sellers.

Free and open-source code does not only express the free choices of its authors, but it provides practical ways to expand the freedoms of those who use and depend on that code. Before I read Alan's piece, I didn't see the connection between proprietary code and a larger paradigm anchored in persuasion. The problem with proprietary code in many cases begins with motivation. It's not just about making secret bits and driving up sale value by making those bits scarce. It's about controlling the customer. The persuasion paradigm puts a premium on captive, dependent customers. Worse, it restricts the usefulness of products to just what works for the seller. This usually involves limiting choices for customers.

What Alan calls the personal decision-making paradigm has been supported from Day Zero in the Free and Open-Source Development world. What we have here, I submit, is a usefulness paradigm—one that has had a quiet and transformative influence on all it supports. Today, that includes Linux, the Internet and countless other useful goods, all of which support far more economic vitality and growth than we ever enjoyed from the persuasion paradigm. (With due credit for all it did achieve in its day.)

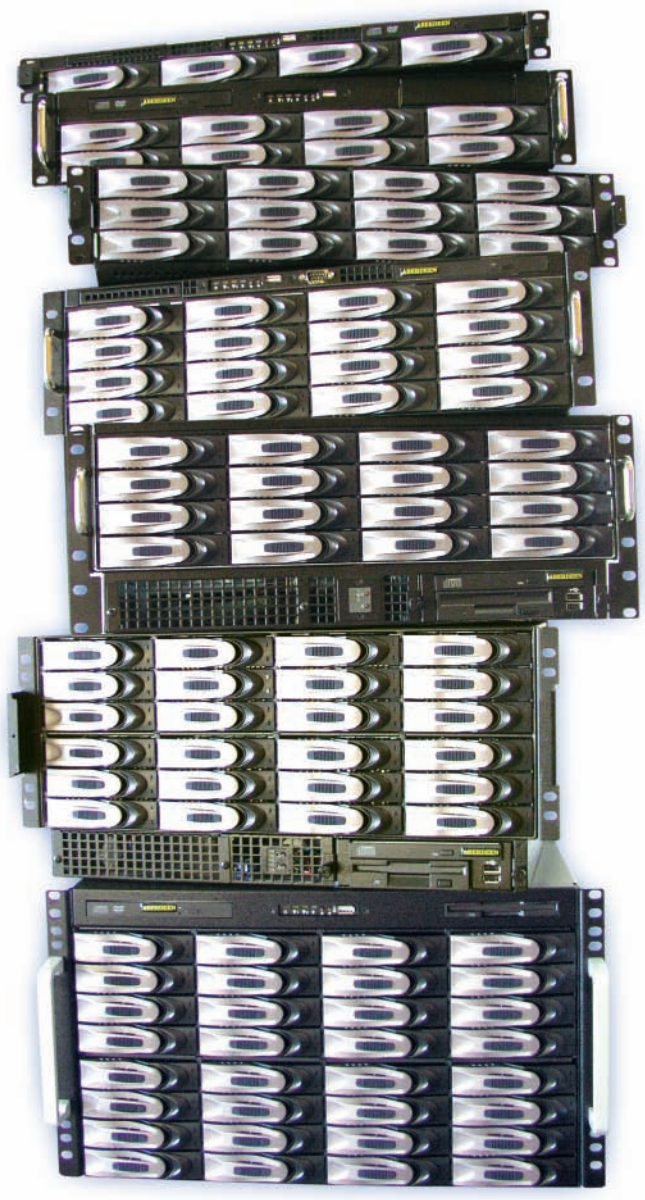
I share that observation with the intent that it will prove useful. If it's not, we'll patch it, discard it or rewrite it. If it is, any of us are still free to improve it.■

Doc Searls is Senior Editor of *Linux Journal*. He is also a Visiting Scholar at the University of California at Santa Barbara and a Fellow with the Berkman Center for Internet and Society at Harvard University.

The Straight Talk People℠
SINCE 1991

ABERDEEN

HERE IS HOW THE BEST STACK UP



4-DRIVE 1U ABERNAS

Up to 4TB Capacity
- Intel® Pentium® 4 Processor
- 1GB DDR Memory
- Available as Linux or Windows NAS
- 300W Power Supply
- From 1TB to 4TB

Starting at **$2,795**

8-DRIVE 2U ABERNAS

Up to 8TB Capacity
- Single Dual-Core Intel Xeon® Processor
- 2GB DDR2 Memory
- Available as Linux or Windows NAS
- 500W Redundant Power
- From 2TB to 8TB

Starting at **$4,295**

12-DRIVE 2U ABERNAS

Up to 12TB Capacity
- Single Dual-Core Intel Xeon Processor
- 2GB DDR2 Memory
- Available as Linux or Windows NAS
- 500W Redundant Power
- From 9TB to 12TB

Starting at **$8,495**

16-DRIVE 3U ABERNAS

Up to 16TB Capacity
- Dual Quad-Core Intel Xeon Processors
- 2GB DDR2 Memory
- Windows Storage Server 2003
- 500W Redundant Power
- From 8TB to 16TB

Starting at **$8,795**

24-DRIVE 5U ABERNAS

Up to 24TB Capacity
- Dual Quad-Core Intel Xeon Processors
- 2GB DDR2 Memory
- Windows Storage Server 2003
- 950W Redundant Power
- From 12TB to 24TB

Starting at **$11,795**

32-DRIVE 6U ABERNAS

Up to 32TB Capacity
- Dual Quad-Core Intel Xeon Processors
- 2GB DDR2 Memory
- Windows Storage Server 2003
- 1350W Redundant Power
- From 16TB to 32TB

Starting at **$14,795**

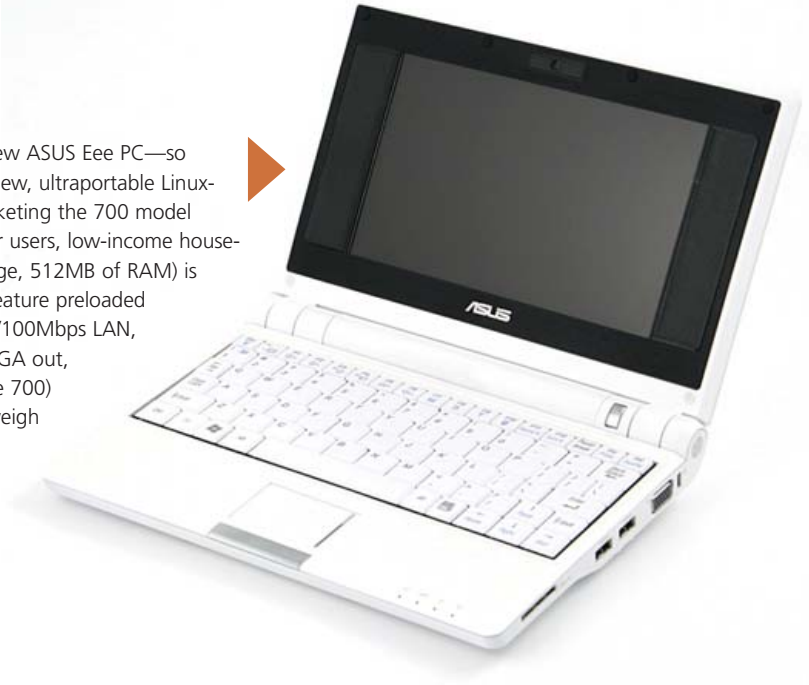
Features & Benefits:
- Up to 32TB of Storage Capacity
- Available in three Operating Systems
  - GUI-Based Linux featuring iSCSI
  - Windows Storage Server 2003 R2
  - Windows Unified Data Storage Server w/iSCSI
- Optimized SMB and NFS Performance
- Internal Windows OS Drive
- Linux OS on DOM
- Recovery from DVD or USB
- RAID 6 w/automatic rebuild
- Multi-platform File Sharing — Windows, Linux, Unix, FTP, Macintosh

"Aberdeen surpasses HP … markedly higher scores … AberNAS 128 boasts outstanding features"
Network Computing—Aberdeen AberNAS 128

intel
Xeon® inside™
Quad-core. Unmatched.

888-297-7409
www.aberdeeninc.com/lj021

ASUS Eee PC 700/701



Fear not, gadget geeks, for in future issues we'll fully cover the new ASUS Eee PC—so much so that you'll grumble "enough already!" The Eee PC is a new, ultraportable Linux-based laptop starting at (this is not a typo) $259 US. ASUS is marketing the 700 model (2GB Flash storage, 256MB of RAM) to first-time/elderly computer users, low-income households and K–12 education, while the 701 model (4GB Flash storage, 512MB of RAM) is mainly for PC owners seeking convenient mobility. Both models feature preloaded Xandros Linux, Intel Celeron-M 900MHz processor, 7" display, 10/100Mbps LAN, 802.11 b/g wireless, three USB 2.0 ports, MMC/SD card reader, VGA out, Windows XP compatibility/drivers, built-in camera (optional on the 700) and a four-cell battery. There are no optical drives; both models weigh in at 0.9kg/2lbs. At the time of this writing, ASUS projects that dealers should have the Eee PC by late September 2007.

www.asuseeepc.com

Lenovo's ThinkPad T Series Linux Notebooks

A year ago we saw Lenovo tepidly dip its pinky-toe into the Linux waters by certifying and supporting its ThinkPad T60p notebook with user-installed SUSE Linux Enterprise Desktop (SLED). This year finds Lenovo frolicking in the balmy Linux Sea, offering SLED preloaded and supported on its business-oriented ThinkPad T Series. Lenovo says that strong demand for Linux notebooks from customers made the preload option inevitable. Preloaded ThinkPads will be available to the general public in Q4 2007.

www.lenovo.com/linux

Storix, Inc.'s SBAdmin

Storix, Inc., recently released SBAdmin v6.2, touted as the first backup and system recovery solution to integrate with IBM's Tivoli Storage Manager (TSM). SBAdmin complements TSM's features and capabilities for Linux and AIX systems. SBAdmin also writes directly to the TSM server, which cuts extraneous steps, saves time and increases reliability. In addition, the application provides TSM users with disaster-recovery capabilities, including a feature called Adaptable System Recovery, which enables a system restore to the same or dissimilar hardware.

www.storix.com

MOTOROLA

Motorola's MOTOMAGX Mobile Linux Platform

In the near future, Motorola expects 60% of its handsets to run on Linux, and the new MOTOMAGX platform is a means for realizing that goal. Motorola calls MOTOMAGX its "next-generation mobile Linux platform" that will "deliver new levels of openness, flexibility and support for third-party applications" on its devices. The company also hopes to empower its developer community to innovate in exciting ways. Currently, MOTOMAGX supports apps developed in Java ME and will soon support WebUI and native Linux application environments.

www.motorola.com

Olive's OPUS 307S

Who knows where you'll find Linux next? One sure spot is the embedded brain inside Olive Media Products' new OPUS 307S audio system. The OPUS 307S, part of Olive's OPUS N°3 product line, is a stylish audio component that goes in your stereo rack and contains all of your digital music on its 250GB hard drive. The OPUS can store up to 700 CDs in lossless quality. Improvements on the OPUS 307S over previous Olive devices include perpendicular recording technology for lower power consumption, higher storage density, improved long-term reliability and inaudible operation. In addition, Olive will preload up to 300 of your CDs for free.

www.olive.us

Lantronix's SecureLinx Branch Office Manager

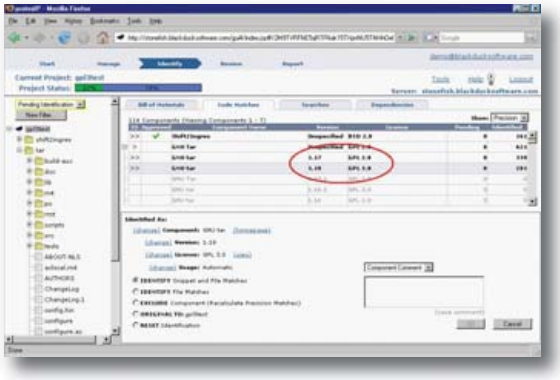
Move over Swiss Army knife, Lantronix has released its new SecureLinx Branch (SLB) Office Manager product, an IT management appliance that integrates a console server, power management and an Ethernet switch into a 1U rack-mountable device. With SLB, system administrators can securely manage a diverse range of servers and IT infrastructure equipment in branch/remote offices from anywhere via the Internet. SLB allows enterprises to avoid the cost of dedicated technicians and other expensive overhead at each satellite location.

www.lantronix.com

Black Duck Software's protexIP

Software development with open-source code is great, but it can be complicated. This is why Black Duck Software created protexIP/development (now v4.4), "a platform that helps companies govern how their software assets are created, managed and licensed." protexIP helps developers and legal counsel in managing the use of code from open-source projects who have explicitly decided to switch to GPLv3 and those that have not. Also included is an enhanced KnowledgeBase, a library of open-source and vendor-added code software components with detailed licensing information for more than 140,000 components.

www.blackducksoftware.com



Sendio's I.C.E Box E-Mail Integrity Service Appliance

We've been tracking the buzz surrounding Sendio's I.C.E Box, an e-mail security appliance that reputedly blocks 100% of spam with no false positives. Although the I.C.E. Box has been shipping for several months, its Linux credentials recently came to our attention. Shunning the antispam filter approach, the I.C.E. Box performs a one-time verification that the sender of the message is someone with whom you indeed wish to communicate. The result is an end to the junk while maintaining legitimate communications. The appliance integrates seamlessly with any e-mail server and LDAP environment. A built-in Kaspersky antivirus engine also is included.

www.sendio.com

Trinity's Digital Audio Workstation

Here is an audio device that is not brand-new but new to me and hopefully to you too—Trinity Audio Group's Trinity Digital Audio Workstation. This slick, all-in-one device is a portable, professional recording studio that allows one to do anything imaginable with linear audio (for example, sample, edit, mix, play back and so on). Trinity does everything your laptop with an audio interface can do, only more conveniently and portably. Trinity runs on Linux and works with WAV, MP3 and Ogg Vorbis files. Several core audio apps are included, such as Audacity, Hydrogen drum machine, Ardour and more. Built-in 802.11g Wi-Fi lets you podcast from the field.

www.trinityaudiogroup.com

IBM's Big Green Linux Initiative

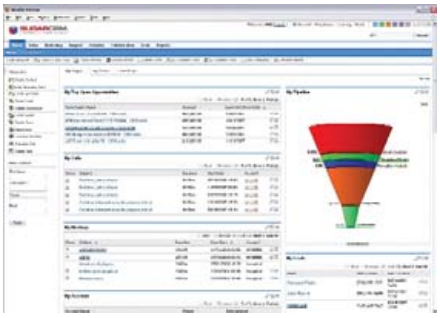
Thankfully, firms like IBM are starting to look at our serious environmental problems as challenges rather than barriers. IBM's new Big Green Linux initiative seeks to leverage Linux and other technologies to reduce costs and energy consumption by building cooler data centers for itself and its customers. Big Green Linux is a subset of IBM's broader, yet similar, Project Big Green. Specifics of Big Green Linux include server consolidation to System x and System p platforms; new efficient products, such as the Information Server Blade; contributions to the Linux kernel (such as tickless low-power state) and others.

www.ibm.com/linux

SugarCRM

By the time you flip to this page, SugarCRM will have its sweet new version 5.0 ready for you. This is the ninth major release of the popular commercial open-source customer relationship management package. New features, such as the ability to build custom modules, a new Ajax e-mail client and a Multi-Instance On-Demand architecture, are said to "enhance the ability to build, customize, scale and upgrade the application". SugarCRM says its innovations are derived from feedback from 2,000 commercial customers and thousands of Open Source community members.

www.sugarcrm.com



Trusted Computer Solutions' Security Blanket

Trusted Computer Solutions has long made security solutions for the CIA and other spy agencies in Washington, DC, and now its new product, Security Blanket, aims to harden your Linux system with the same fervor. Trusted Computer Solutions calls Security Blanket "a system lock-down and security management tool that enables systems administrators to configure and enhance the security level of the Red Hat Enterprise Linux operating platform automatically". The product's advantage is to simplify the current arduous methods (such as using Bastille) for "hardening" systems. Its menu-driven UI allows one to run either customized profiles or predefined ones that automate industry-standard best practices from the National Institute of Standards and the Center for Internet Security. Security Blanket supports Red Hat Enterprise Linux 4 and 5.

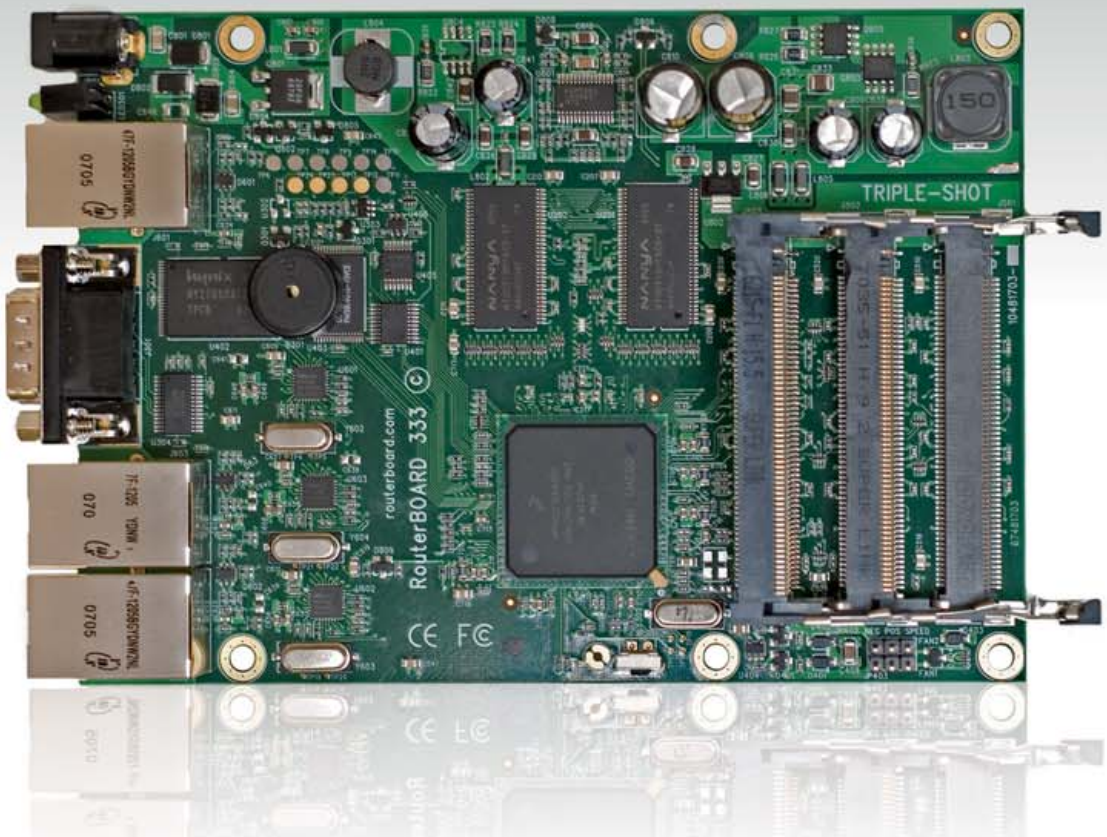
www.trustedcs.com

Please send information about releases of Linux-related products to James Gray at newproducts@linuxjournal.com or New Products c/o *Linux Journal*, 1752 NW Market Street, #200, Seattle, WA 98107. Submissions are edited for length and content.

RouterBOARD™ 333

www.mikrotik.com

Triple-Shot AP/Firewall/Routing $180
PowerPC E300 266/333MHz CPU
QUICC Engine 175MHz Coprocessor
Up to 81,000pps, Full duplex 3xFE throughput
64MB DDR RAM
Onboard NAND storage
Three 10/100 ethernet ports MDI/X
Three miniPCI Type IIIA/IIIB slots
PoE/Jack: 9-28V DC; Overvoltage / polarity protection
19W available to miniPCI cards
Input Voltage monitor
150mm x 105mm (5.9in x 4.13in)



Red Hat Enterprise Linux Cluster Suite

Building a highly available solution using the RHEL cluster suite.

KHURRAM SHIRAZ

When mission-critical applications fail, so does your business. This often is a true statement in today's environments, where most organizations spend millions of dollars making their services available 24/7, 365 days a year. Organizations, regardless of whether they are serving external customers or internal customers, are deploying highly available solutions to make their applications highly available.

In view of this growing demand, almost every IT vendor currently is providing high-availability solutions for its specific platform. Famous commercial high-availability solutions include IBM's HACMP, Veritas' Cluster Server and HP's Serviceguard.

If you're looking for a commercial high-availability solution on Red Hat Enterprise Linux, the best choice probably is the Red Hat Cluster Suite.

In early 2002, Red Hat introduced the first member of its Red Hat Enterprise Linux family of products, Red Hat Enterprise Linux AS (originally called Red Hat Linux Advanced Server). Since then, the family of products has grown steadily, and it now includes Red Hat Enterprise Linux ES (for entry- and mid-range servers) and Red Hat Enterprise Linux WS (for desktops/workstations). These products are designed specifically for use in enterprise environments to deliver superior application support, performance, availability and scalability.

The original release of Red Hat Enterprise Linux AS version 2.1 included a high-availability clustering feature as part of the base product. This feature was not included in the smaller Red Hat Enterprise Linux ES product. However, with the success of the Red Hat Enterprise Linux family, it became clear that high-availability clustering was a feature that should be made available for both AS and ES server products. Consequently, with the release of Red Hat Enterprise Linux version 3 in October 2003, the high-availability clustering feature was packaged into an optional layered product called the Red Hat Cluster Suite, and it was certified for use on both the Enterprise Linux AS and Enterprise Linux ES products.

The RHEL cluster suite is a separately licensed product and can be purchased from Red Hat on top of Red Hat's base ES Linux license.

Red Hat Cluster Suite Overview

The Red Hat Cluster Suite has two major features. One is the Cluster Manager that provides high availability, and the other feature is called IP load balancing (originally called Piranha). The Cluster Manager and IP load balancing are complementary high-availability technologies that can be used separately or in combination, depending on application requirements. Both of these technologies are integrated in Red Hat's Cluster Suite. In this article, I focus on the Cluster Manager. Table 1 shows the major components of the RHEL Cluster Manager.

Table 1. RHEL Cluster Manager Components

SOFTWARE SUBSYSTEM	COMPONENT	PURPOSE
Fence	fenced	Provides fencing infrastructure for specific hardware platforms.
DLM	libdlm, dlm-kernel	Contains distributed lock management (DLM) library.
CMAN	cman	Contains the Cluster Manager (CMAN), which is used for managing cluster membership, messaging and notification.
GFS and related locks	Lock_NoLock	Contains shared filesystem support that can be mounted on multiple nodes concurrently.
GULM	gulm	Contains the GULM lock management user-space tools and libraries (an alternative to using CMAN and DLM).
Rgmanager	clurgmgrd, clustat	Manages cluster services and resources.
CCS	ccsd, ccs_test and ccs_tool	Contains the cluster configuration services dæmon (ccsd) and associated files.
Cluster Configuration Tool	System-config-cluster	Contains the Cluster Configuration Tool, used to configure the cluster and display the current status of the nodes, resources, fencing agents and cluster services graphically.
Magma	magma and magma-plugins	Contains an interface library for cluster lock management and required plugins.
IDDEV	iddev	Contains the libraries used to identify the filesystem (or volume manager) in which a device is formatted.

Shared Storage and Data Integrity

Lock management is a common cluster infrastructure service that provides a mechanism for other cluster infrastructure components to synchronize their access to shared resources. In a Red Hat cluster, DLM (Distributed Lock Manager) or, alternatively, GULM (Grand Unified Lock Manager) are possible lock manager choices. GULM is a server-based unified cluster/lock manager for GFS, GNBD and CLVM. It can be used in place of CMAN and DLM. A single GULM server can be run in standalone mode but introduces a single point of failure for GFS. Three or five GULM servers also can be run together, in which case the failure of one or two servers can be tolerated, respectively. GULM servers usually are run on dedicated machines, although this is not a strict requirement.

In my cluster implementation, I used DLM, and it runs in each cluster node. DLM is good choice for small clusters (up to two nodes), because it removes quorum requirements as imposed by the GULM mechanism).

Based on DLM or GLM locking functionality, there are two basic techniques that can be used by the RHEL cluster for ensuring data integrity in concurrent access environments. The traditional way is the use of CLVM, which works well in most RHEL cluster implementations with LVM-based logical volumes.

Another technique is GFS. GFS is a cluster filesystem that allows a cluster of nodes to access simultaneously a block device that is shared among the nodes. It employs distributed metadata and multiple journals for optimal operation in a cluster. To maintain filesystem integrity, GFS uses a lock manager (DLM or GULM) to coordinate I/O. When one node changes data on a GFS filesystem, that change is visible immediately to the other cluster nodes using that filesystem.

Hence, when you are implementing a RHEL cluster with concurrent data access requirements (such as, in the case of an Oracle RAC implementation), you can use either GFS or CLVM. In most Red Hat cluster implementations, GFS is used with a direct access configuration to shared SAN from all cluster nodes. However, for the same purpose, you also can deploy GFS in a cluster that is connected to a LAN with servers that use GNBD (Global Network Block Device) or two iSCSI (Internet Small Computer System Interface) devices.

Both GFS and CLVM use locks from the lock manager. However, GFS uses locks from the lock manager to synchronize access to filesystem metadata (on shared storage), while CLVM uses locks from the lock manager to synchronize updates to LVM volumes and volume groups (also on shared storage).

For nonconcurrent RHEL cluster implementations, you can rely on CLVM, or you can use native RHEL journaling-based techniques (such as ext2 and ext3). For nonconcurrent access clusters, data integrity issues are minimal; I tried to keep my cluster implementations simple by using native RHEL OS techniques.

Fencing Infrastructure

Fencing also is an important component of every RHEL-based cluster implementation. The main purpose of the fencing implementation is to ensure data integrity in a clustered environment.

In fact, to ensure data integrity, only one node can run a cluster service and access cluster service data at a time. The use

of power switches in the cluster hardware configuration enables a node to power-cycle another node before restarting that node's cluster services during the failover process. This prevents any two systems from simultaneously accessing the same data and corrupting it. It is strongly recommended that fence devices (hardware or software solutions that remotely power, shut down and reboot cluster nodes) are used to guarantee data integrity under all failure conditions. Software-based watchdog timers are an alternative used to ensure correct operation of cluster service failover; however, in most RHEL cluster implementations, hardware fence devices are used, such as HP ILO, APC power switches, IBM BladeCenter devices and the Bull NovaScale Platform Administration Processor (PAP) Interface.

Note that for RHEL cluster solutions with shared storage, an implementation of the fence infrastructure is a mandatory requirement.

Step-by-Step Implementation of a RHEL Cluster

Implementation of RHEL clusters starts with the selection of proper hardware and connectivity. In most implementations (without IP load balancing), shared storage is used with two, or more than two, servers running the RHEL operating system and RHEL cluster suite.

A properly designed cluster, whether you are building a RHEL-based cluster or an IBM HACMP-based cluster, should not contain any single point of failure. Keeping this in mind, you have to remove any single point of failure from your cluster design. For this purpose, you can place your servers physically in two separate racks with redundant power supplies. You also have to remove any single point of failure from the network infrastructure used for the cluster. Ideally, you should have at least two network adapters on each cluster node, and two network switches should be used for building the network infrastructure for the cluster implementation.

Software Installation

Building a RHEL cluster starts with the installation of RHEL on two cluster nodes. My setup has two HP Proliant servers (DL740) with shared fiber storage (HP MSA1000 storage). I started with a RHEL v4 installation on both nodes. It's best to install the latest available operating system version and its updates. I selected v4 update 4 (which was the latest version of RHEL when I was building that cluster). If you have a valid software subscription from Red Hat, you can log in to the Red Hat network, and go to software channels to download the latest update available. Later, once you download the ISO images, you can burn it to CDs using any appropriate software. During the RHEL OS installation, you will go through various configuration selections, the most important of which are the date and time-zone configuration, the root user password setting, firewall settings and OS security level selection. Another important configuration option is network settings. Configuration of these settings can be left for a later stage, especially in building a high-availability solution with Ether-channel (or Ethernet bonding configuration).

You may need to install additional drivers after you install the OS. In my case, I downloaded the RHEL support package for the DL740 servers (the HP Proliant support pack, which is available from **h18004.www1.hp.com/products/servers/linux/dl740-drivers-cert.html**).

The next step is installing the cluster software package itself. This package, again, is available from the RHEL network, and you definitely have to select the latest available cluster package. I selected rhel-cluster-2.4.0.1 for my setup, which was the latest cluster suite available at the time.

Once downloaded, the package will be in tar format. Extract it, and then install at least the following RPMs, so that the RHEL cluster with DLM can be installed and configured:

- Magma and magma-plugins
- Perl-net-telnet
- Rgmanager
- System-config-cluster
- DLM and dlm-kernel
- DLM-kernel-hugemem and SMP support for DLM
- Iddev and ipvsadm
- Cman, cman-smp, cman-hugemem and cman-kernelheaders
- Ccs

Restart both RHEL cluster nodes after installing vendor-related hardware support drivers and the RHEL cluster suite.

Network Configuration

For network configuration, the best way to proceed is to use the network configuration GUI. However, if you plan to use Ethernet channel bonding, the configuration steps are slightly different.

Ethernet channel bonding allows for a fault-tolerant network connection by combining two Ethernet devices into one virtual device. The resulting channel-bonded interface ensures that if one Ethernet device fails, the other device will become active. Ideally, connections from these Ethernet devices should go to separate Ethernet switches or hubs, so that the single point of failure is eliminated, even on the Ethernet switch and hub level.

To configure two network devices for channel bonding, perform the following on node 1:

1) Create bonding devices in /etc/modules.conf. For example, I used the following commands on each cluster node:

```
alias bond0 bonding
options bonding miimon=100 mode=1
```

Doing this loads the bonding device with the bond0 interface name and passes options to the bonding driver to configure it as an active-backup master device for the enslaved network interfaces.

2) Edit the /etc/sysconfig/network-scripts/ifcfg-eth0 configuration file for eth0 and the /etc/sysconfig/network-scripts/ifcfg-eth1 file for the eth1 interface, so that these files show identical contents, as shown below:

```
DEVICE=ethx
USERCTL= no
ONBOOT=yes
MASTER=bond0
SLAVE=yes
BOOTPROTO=none
```

This enslaves ethX (replace X with the assigned number of the

Ethernet devices) to the bond0 master device.

3) Create a network script for the bonding device (for example, /etc/sysconfig/network-scripts/ifcfg-bond0), which would appear like the following example:

```
DEVICE=bond0
USERCTL=no
ONBOOT=yes
BROADCAST=172.16.2.255
NETWORK=172.16.2.0
NETMASK=255.255.255.0
GATEWAY=172.16.2.1
IPADDR=172.16.2.182
```

4) Reboot the system for the changes to take effect.

5) Similarly, on node 2, repeat the same steps with the only difference being that the file /etc/sysconfig/network-scripts/ifcfg-bond0 should contain an IPADDR entry with the value of 172.16.2.183.

As a result of these configuration steps, you will end up with two RHEL cluster nodes with IP addresses of 172.16.2.182 and 172.16.2.183, which have been assigned to virtual Ethernet channels (the underlying two physical Ethernet adapters for each Ethernet channel).

Now, you easily can use the network configuration GUI on the cluster nodes to set other network configuration details, such as hostname and primary/secondary DNS server configuration. I set Commsvr1 and Commsvr2 as the hostnames for the cluster nodes and also ensured that names resolution in both long names and short names would work fine from both the DNS server and the /etc/hosts file.

A RHEL cluster, by default, uses /etc/hosts for node name resolution. The cluster node name needs to match the output of `uname -n` or the value of HOSTNAME in /etc/sysconfig/network.

If you have an additional Ethernet interface in each cluster node, it is always a good idea to configure a separate IP network as an additional network for heartbeats between cluster nodes. It is important that the RHEL cluster uses, by default, eth0 on the cluster nodes for heartbeats. However, it is still possible to use other interfaces for additional heartbeat exchanges.

For this type of configuration, you simply can use the network configuration GUI to assign IP addresses—for example, 192.168.10.1 and 192.168.10.2 on eth2, and get it resolved from the /etc/hosts file.

Listing 1. Contents of the /etc/hosts File on Each Server

```
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1       localhost.localdomain   localhost
172.16.2.182    Commsvr1        Commsvr1.kmefic.com.kw
172.16.2.183    Commsvr2
172.16.1.186    Commilo1        Commilo1.kmefic.com.kw
172.16.1.187    Commilo2        Commilo2.kmefic.com.kw
172.16.2.188    Commserver
192.168.10.1    node1
192.168.10.2    node2
172.16.2.4      KMETSM
```

Setup of the Fencing Device

As I was using HP hardware, I relied on the configuration of the HP ILO devices as a fencing device for my cluster. However, you may consider configuring other fencing devices, depending on the hardware type used for your cluster configuration.

To configure HP ILO, you have to reboot your servers and press the F8 key to enter into the ILO configuration menus. Basic configuration is relatively simple; you have to assign IP addresses to ILO devices with the name of the ILO device. I assigned 172.16.1.100 with Commilo1 as the name of ILO device on node1, and 172.16.1.101 with Commilo2 as the ILO device name on node2. Be sure, however, to connect Ethernet cables to the ILO adapters, which usually are marked clearly on the back side of HP servers.

Once rebooted, you can use the browsers on your Linux servers to access ILO devices. The default user name is Administrator, with a password that usually is available on the hard-copy tag associated with the HP servers. Later, you can change the Administrator password to a password of your choice, using the same Web-based ILO administration interface.

Setup of the Shared Storage Drive and Quorum Partitions

In my cluster setup environment, I used an HP fiber-based shared storage MSA1000. I configured a RAID-1 of 73.5GB using the HP smart array utility, and then assigned it to both of my cluster nodes using the selective host presentation feature.

After rebooting both nodes, I used HP fiber utilities, such as hp_scan, so that both servers should be able to see this array physically.

To verify the physical availability of shared storage for both cluster nodes, look in the /dev/proc/proc file for an entry like /dev/sda or /dev/sdb, depending upon your environment.

Once you find your shared storage on the OS level, partition it according to your cluster storage requirements. I used the parted tool on one of my cluster nodes to partition the shared storage. I created two small primary partitions to hold raw devices, and a third primary partition was created to hold the shared data filesystem:

```
Parted> select /dev/sda

Parted > mklabel /dev/sda msdos

Parted > mkpart primary ext3 0 20

Parted > mkpart primary ext3 20 40

Parted > mkpart primary ext3 40 40000
```

I rebooted both cluster nodes and created the /etc/sysconfig/rawdevices file with the following contents:

```
/dev/raw/raw1           /dev/sda1
/dev/raw/raw2           /dev/sda2
```

A restart of rawdevices services on both nodes will configure raw devices as quorum partitions:

```
/home/root> services rawdevices restart
```

I then created a JFS2 filesystem on the third primary partition using the `mke2jfs` command; however, its related entry should not be put in

It is strongly recommended that fence devices (hardware or software solutions that remotely power, shut down and reboot cluster nodes) are used to guarantee data integrity under all failure conditions.

the /etc/fstab file on either cluster node, as this shared filesystem will be under the control of the Rgmanager of the cluster suite:

```
/home/root> mke2jfs -j -b 4096 /dev/sda3
```

Now, you can create a directory structure called /shared/data on both nodes and verify the accessibility of the shared filesystem from both cluster nodes by mounting that filesystem one by one at each cluster node (`mount /dev/sda3 /shared/data`). However, never try to mount this filesystem on both cluster nodes simultaneously, as it might corrupt the filesystem itself.

Cluster Configuration

Almost everything required for cluster infrastructure has been done, so the next step is configuring the cluster itself.

A RHEL cluster can be configured in many ways. However, the easiest way to configure a RHEL cluster is to use the RHEL GUI and go to System Management→Cluster Management→Create a cluster.

I created a cluster with the cluster name of Commcluster, and with node names of Commsvr1 and Commsvr2. I added fencing to both nodes—fencing devices Commilo1 and Commilo2, respectively—so that each node would have one fence level with one fence device. If you have multiple fence devices in your environment, you can add another fence level with more fence devices to each node.

I also added a shared IP address of 172.16.2.188, which will be used as the service IP address for this cluster. This is the IP address that also should be used as the service IP address for applications or databases (like for listener configuration, if you are going to use an Oracle database in the cluster).

I added a failover domain, namely Kmeficfailover, with priorities given in the following sequence:

```
Commsvr1
Commsvr2
```

I added a service called CommSvc and then put that service in the above-defined failover domain. The next step is adding resources to this service. I added a private resource of the filesystem type, which has the characteristic of device=/dev/sd3, mountpoint of /shared/data and mount type of ext3.

I also added a private resource of the script type (/root/CommS.sh) to service CommSvc. This script will start my C-based application, and therefore, it has to be present in the /root directory on both cluster nodes. It is very important to have correct ownership of root and security; otherwise, you can expect unpredictable behavior during cluster startup and shutdown.

Application or database startup and shutdown scripts are very important for a RHEL-based cluster to function properly. RHEL clusters use the same scripts for providing application/database monitoring and high availability, so every application script used in a RHEL cluster should have a specific format.

All such scripts should at least have start and stop subsections, along with a status subsection. When an application or database is available and running, the status subsection of the script should return a value of 0, and when an application is not running or available, it should return a value of 1. The script also should contain a restart subsection, which tries to restart services if the application is found to be dead.

A RHEL cluster always tries to restart the application on the same node that was the previous owner of the application, before trying to move that application to the other cluster node. A sample application script, which was used in my RHEL cluster implementation (to provide

Listing 2. Sample Application Script

```
#Script Name: CommS.sh
#Script Purpose: To provide application
#start/stop/status under Cluster
#Script Author: Khurram Shiraz

#!/bin/sh
basedir=/home/kmefic/KMEFIC/CommunicationServer
case $1 in
'start')
cd $basedir
su kmefic -c "./CommunicationServer -f Dev-CommunicationServer.conf"
exit 0
;;
'stop')
z=`ps -ef | grep Dev-CommunicationServer | grep -v "grep"|
 ➥awk ' { print $2 } '
`
if [[ $? -eq 0 ]]
then
kill -9 $z
fuser -mk /home/kmefic
exit 0
fi
;;
'restart')
   /root/CommunicationS.sh stop
   sleep 2
  echo Now starting......
  /root/CommunicationS.sh start
    echo "restarted"
    ;;

'status')
ps -U kmefic | grep CommunicationSe 1>/dev/null
if [[ $? = 0 ]]
then
exit 0
else
exit 1
fi
;;
esac
```

high availability to a legacy C-based application) is shown in Listing 2.

Finally, you have to add a shared IP address (172.16.2.188) to the service present in your failover domain, so that the service should contain three resources: two private resources (one filesystem and one script) and one shared resource, which is the service IP address for the cluster.

The last step is synchronizing the cluster configuration across the cluster nodes. The RHEL cluster administration and configuration tool provides a "save configuration to cluster" option, which will appear once you start the cluster services. Hence, for the first synchronization, it is better to send the cluster configuration file manually to all cluster nodes. You easily can use the scp command to synchronize the /etc/cluster/cluster.conf file across the cluster nodes:

```
/home/root> scp /etc/cluster/cluster.conf Commsvr2:/etc/cluster/cluster.conf
```

Once synchronized, you can start cluster services on both cluster nodes. You should start and stop RHEL-related cluster services, in sequence.

To start:

```
service ccsd start
service cman start
service fenced start
service rgmanager start
```

To stop:

```
service rgmanager stop
service fenced stop
service cman stop
service ccsd stop
```

If you use GFS, startup/shutdown of the gfs and clvmd services have to be included in this sequence.

Additional Considerations

In my environment, I decided not to start cluster services at RHEL boot time and not to shut down these services automatically when shutting down the RHEL box. However, if your business requires 24/7 service availability, you can do this easily by using the chkconfig command.

Another consideration is logging cluster messages in a different log file. By default, all cluster messages go into the RHEL log messages file (/var/log/messages), which makes cluster troubleshooting somewhat difficult in some scenarios. For this purpose, I edited the /etc/syslog.conf file to enable the cluster to log events to a file that is different from the default log file and added the following line:

```
daemon.* /var/log/cluster
```

To apply this change, I restarted syslogd with the service syslog restart command. Another important step is to specify the time period for rotating cluster log files. This can be done by specifying the name of the cluster log file in the /etc/logrotate.conf file (the default is a weekly rotation):

```
/var/log/messages /var/log/secure /var/log/maillog /var/log/spooler
/var/log/boot.log /var/log/cron /var/log/cluster {
    sharedscripts postrotate
          /bin/kill -HUP `cat /var/run/syslogd.pid 2> /dev/null` 2>
           /dev/null || true
    endscript
}
```

You also have to pay special attention to keeping UIDs and GIDs synchronized across cluster nodes. This is important in making sure proper permissions are maintained, especially with reference to the shared data filesystem.

GRUB also needs to conform to the suite environment's specific needs. For instance, many system administrators, in a RHEL cluster environment, reduce the GRUB selection timeout to some lower values, such as two seconds, to accelerate system restart time.

Database Integration with a RHEL Cluster

The same RHEL cluster infrastructure can be used for providing high availability to databases, such as Oracle, MySQL and IBM DB2.

The most important thing to remember is to base your database-related services on a shared IP address—for example, you have to configure Oracle listener based on the shared service IP address.

Next, I explain, in simple steps, how to use an already-configured RHEL cluster to provide high availability to a MySQL database server, which is, no doubt, one of the most commonly used databases on RHEL.

I assume that the MySQL-related RPMs are installed on both cluster nodes and that the RHEL cluster already is configured with a service IP address of 172.16.2.188.

Now, you simply need to define a failover domain using the cluster configuration tool (with the cluster node of your choice having a higher priority). This failover domain will have the MySQL service, which, in turn, will have two private resources and one shared resource (the service IP address).

One of the private resources should be of the filesystem type (in my configuration, it has a mountpoint of /shared/mysqld), and the other private resource should be of the script type, pointing toward the /etc/init.d/mysql.server script. The contents of this script, which should be available on both cluster nodes, is shown in Listing 3 on the *LJ* FTP site at **ftp.linuxjournal.com/pub/lj/listings/issue163/9759.tgz**.

This script sets the data directory to /shared/mysqld/data, which is available on our shared RAID array and should be available from both cluster nodes.

Testing for high availability of the MySQL database can be done easily with the help of any MySQL client. I used SQLyog, which is a Windows-based MySQL client. I connected to the MySQL database on Commsvr1 and then crashed this cluster node using the halt command. As a result of this system crash, the RHEL cluster events were triggered, and the MySQL database automatically restarted on Commsvr2. This whole failover process took one to two minutes and happened quite seamlessly.

Summary

RHEL clustering technology provides a reliable high-available infrastructure that can be used for meeting 24/7 business requirements for databases as well as legacy applications. The most important thing to remember is that it is best to plan carefully before the actual implementation and test your cluster and all possible failover scenarios thoroughly before going live with a RHEL cluster. A well-documented cluster test plan also can be helpful in this regard.■

Khurram Shiraz is senior system administrator at KMEFIC, Kuwait. In his eight years of IT experience, he has worked mainly with IBM technologies and products, especially AIX, HACMP Clustering, Tivoli and IBM SAN/NAS storage. He also has worked with the IBM Integrated Technology Services group. His areas of expertise include design and implementation of high-availability and DR solutions based on pSeries, Linux and Windows infrastructures. He can be reached at aix_tiger@yahoo.com.

Getting Started with Heartbeat

Your first step toward high-availability bliss.

Daniel Bartholomew

In every work environment with which I have been involved, certain servers absolutely always must be up and running for the business to keep functioning smoothly. These servers provide services that always need to be available—whether it be a database, DHCP, DNS, file, Web, firewall or mail server.

A cornerstone of any service that always needs be up with no downtime is being able to transfer the service from one system to another gracefully. The magic that makes this happen on Linux is a service called Heartbeat. Heartbeat is the main product of the High-Availability Linux Project.

Heartbeat is very flexible and powerful. In this article, I touch on only basic active/passive clusters with two members, where the active server is providing the services and the passive server is waiting to take over if necessary.

Installing Heartbeat

Debian, Fedora, Gentoo, Mandriva, Red Flag, SUSE, Ubuntu and others have prebuilt packages in their repositories. Check your distribution's main and supplemental repositories for a package named heartbeat-2.

After installing a prebuilt package, you may see a "Heartbeat failure" message. This is normal. After the Heartbeat package is installed, the package manager is trying to start up the Heartbeat service. However, the service does not have a valid configuration yet, so the service fails to start and prints the error message.

You can install Heartbeat manually too. To get the most recent stable version, compiling from source may be necessary. There are a few dependencies, so to prepare on my Ubuntu systems, I first run the following command:

```
sudo apt-get build-dep heartbeat-2
```

Check the Linux-HA Web site for the complete list of dependencies. With the dependencies out of the way, download the latest source tarball and untar it. Use the ConfigureMe script to compile and install Heartbeat. This script makes educated guesses from looking at your environment as to how best to configure and install Heartbeat. It also does everything with one command, like so:

```
sudo ./ConfigureMe install
```

With any luck, you'll walk away for a few minutes, and when you return, Heartbeat will be compiled and installed on every node in your cluster.

Configuring Heartbeat

Heartbeat has three main configuration files:

- /etc/ha.d/authkeys
- /etc/ha.d/ha.cf
- /etc/ha.d/haresources

The authkeys file must be owned by root and be chmod 600. The actual format of the authkeys file is very simple; it's only two lines. There is an auth directive with an associated method ID number, and there is a line that has the authentication method and the key that go with the ID number of the auth directive. There are three supported authentication methods: crc, md5 and sha1. Listing 1 shows an example. You can have more than one authentication method ID, but this is useful only when you are changing authentication methods or keys. Make the key long—it will improve security and you don't have to type in the key ever again.

Listing 1. The /etc/ha.d/authkeys File

```
auth 1
1 sha1 ThisIsAVeryLongAndBoringPassword
```

The ha.cf File

The next file to configure is the ha.cf file—the main Heartbeat configuration file. The contents of this file should be the same on all nodes with a couple of exceptions.

Heartbeat ships with a detailed example file in the documentation directory that is well worth a look. Also, when creating your ha.cf file, the order in which things appear matters. Don't move them around! Two different example ha.cf files are shown in Listings 2 and 3.

The first thing you need to specify is the keepalive—the time between heartbeats in seconds. I generally like to have this set to one or two, but servers under heavy loads might not be able to send heartbeats in a timely manner. So, if you're seeing a lot of warnings about late heartbeats, try increasing the keepalive.

The deadtime is next. This is the time to wait without hearing from a cluster member before the surviving members of the array declare the problem host as being dead.

Next comes the warntime. This setting determines how long to wait before issuing a "late heartbeat" warning.

Sometimes, when all members of a cluster are booted at the same time, there is a significant length of time between when Heartbeat is started and before the network or serial interfaces are ready to send and receive heartbeats. The optional initdead directive takes care of this issue by setting an initial deadtime that applies

Listing 2. The /etc/ha.d/ha.cf File on Briggs & Stratton

```
keepalive 2
deadtime 32
warntime 16
initdead 64
baud 19200
# On briggs the serial device is /dev/ttyS1
# On stratton the serial device is /dev/ttyS0
serial /dev/ttyS1
auto_failback on
node briggs
node stratton
use_logd yes
```

Listing 3. The /etc/ha.d/ha.cf File on Deimos & Phobos

```
keepalive 1
deadtime 10
warntime 5
udpport 694
# deimos' heartbeat ip address is 192.168.1.11
# phobos' heartbeat ip address is 192.168.1.21
ucast eth1 192.168.1.11
auto_failback off
stonith_host deimos wti_nps ares.example.com erisIsTheKey
stonith_host phobos wti_nps ares.example.com erisIsTheKey
node deimos
node phobos
use_logd yes
```

only when Heartbeat is first started.

You can send heartbeats over serial or Ethernet links—either works fine. I like serial for two server clusters that are physically close together, but Ethernet works just as well. The configuration for serial ports is easy; simply specify the baud rate and then the serial device you are using. The serial device is one place where the ha.cf files on each node may differ due to the serial port having different names on each host. If you don't know the tty to which your serial port is assigned, run the following command:

```
setserial -g /dev/ttyS*
```

If anything in the output says "UART: unknown", that device is not a real serial port. If you have several serial ports, experiment to find out which is the correct one.

If you decide to use Ethernet, you have several choices of how to configure things. For simple two-server clusters, ucast (uni-cast) or bcast (broadcast) work well.

The format of the ucast line is:

```
ucast <device> <peer-ip-address>
```

Here is an example:

```
ucast eth1 192.168.1.30
```

If I am using a crossover cable to connect two hosts together, I just broadcast the heartbeat out of the appropriate interface. Here is an example bcast line:

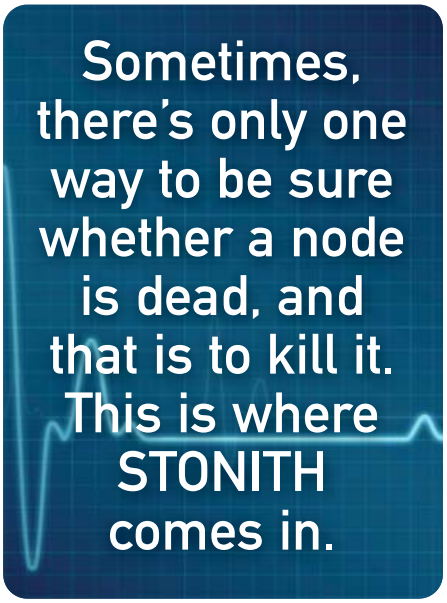
```
bcast eth3
```

There is also a more complicated method called mcast. This method uses multicast to send out heartbeat messages. Check the Heartbeat documentation for full details.

Now that we have Heartbeat transportation all sorted out, we can define auto_failback. You can set auto_failback either to on or off. If set to on and the primary node fails, the secondary node will "failback" to its secondary standby state when the primary node returns. If set to off, when the primary node comes back, it will be the secondary.

It's a toss-up as to which one to use. My thinking is that so long as the servers are identical, if my primary node fails, then the secondary node becomes the primary, and when the prior primary comes back, it becomes the secondary. However, if my secondary server is not as powerful a machine as the primary, similar to how the spare tire in my car is not a "real" tire, I like the primary to become the primary again as soon as it comes back.

Moving on, when Heartbeat thinks a node is dead, that is just a best guess. The "dead" server may still be up. In some cases, if the "dead" server is still partially functional, the consequences are disastrous to the other node members. Sometimes, there's only one way to be sure whether a node is dead, and that is to kill it. This is where STONITH comes in.



STONITH stands for Shoot The Other Node In The Head. STONITH devices are commonly some sort of network power-control device. To see the full list of supported STONITH device types, use the `stonith -L` command, and use `stonith -h` to see how to configure them.

Next, in the ha.cf file, you need to list your nodes. List each one on its own line, like so:

```
node deimos
node phobos
```

The name you use must match the output of `uname -n`.

The last entry in my example ha.cf files is to turn on logging:

```
use_logd yes
```

There are many other options that can't be touched on here. Check the documentation for details.

The haresources File

The third configuration file is the haresources file. Before configuring it, you need to do some housecleaning. Namely, all services that you want Heartbeat to manage must be removed from the system init for all init levels.

On Debian-style distributions, the command is:

```
/usr/sbin/update-rc.d -f <service_name> remove
```

Check your distribution's documentation for how to do the same on your nodes.

Now, you can put the services into the haresources file. As with the other two configuration files for Heartbeat, this one probably won't be very large. Similar to the authkeys file, the haresources file must be *exactly* the same on every node. And, like the ha.cf file, position is *very* important in this file. When control is transferred to a node, the resources listed in the haresources file are started left to right, and when control is transfered to a different node, the resources are stopped right to left. Here's the basic format:

```
<node_name> <resource_1> <resource_2> <resource_3> . . .
```

The node_name is the node you want to be the primary on initial startup of the cluster, and if you turned on auto_failback, this server always will become the primary node whenever it is up. The node name must match the name of one of the nodes listed in the ha.cf file.

Resources are scripts located either in /etc/ha.d/resource.d/ or /etc/init.d/, and if you want to create your own resource scripts, they should conform to LSB-style init scripts like those found in /etc/init.d/. Some of the scripts in the resource.d folder can take arguments, which you can pass using a :: on the resource line. For example, the IPAddr script sets the cluster IP address, which you specify like so:

```
IPAddr::192.168.1.9/24/eth0
```

In the above example, the IPAddr resource is told to set up a cluster IP address of 192.168.1.9 with a 24-bit subnet mask (255.255.255.0) and to bind it to eth0. You can pass other options as well; check the example haresources file that ships with Heartbeat for more information.

Another common resource is Filesystem. This resource is for mounting shared filesystems. Here is an example:

```
Filesystem::/dev/etherd/e1.0::/opt/data::xfs
```

The arguments to the Filesystem resource in the example above are, left to right, the device node (an ATA-over-Ethernet drive in this case), a mountpoint (/opt/data) and the filesystem type (xfs).

For regular init scripts in /etc/init.d/, simply enter them by name. As long as they can be started with `start` and stopped with `stop`, there is a good chance that they will work.

Listings 4 and 5 are haresources files for two of the clusters I run. They are paired with the ha.cf files in Listings 2 and 3, respectively.

The cluster defined in Listings 2 and 4 is very simple, and it has only two resources—a cluster IP address and the Apache 2 Web server. I use this for my personal home Web server cluster. The servers themselves are nothing special—an old PIII tower and a cast-off laptop. The content on the servers is static HTML, and the content is kept in sync with an hourly rsync cron job. I don't trust either "server" very much, but with Heartbeat, I have never had an outage longer than half a second—not bad for two old castaways.

The cluster defined in Listings 3 and 5 is a bit more complicated. This is the NFS cluster I administer at work. This cluster utilizes shared storage in the form of a pair of Coraid SR1521 ATA-over-Ethernet drive arrays, two NFS appliances (also from Coraid) and a STONITH device. STONITH is important for this cluster, because in the event of a failure, I need to be sure that the other device is really dead before mounting the shared storage on the other node. There are five resources managed in this cluster, and to keep the line in haresources from getting too long to be readable, I break it up with line-continuation slashes. If the primary cluster member is having trouble, the secondary cluster kills the primary, takes over the IP address, mounts the shared storage and then starts up NFS. With this cluster, instead of having maintenance issues or other outages lasting several minutes to an hour (or more), outages now don't last beyond a second or two. I can live with that.

Listing 4. A Minimalist haresources File

```
stratton 192.168.1.41 apache2
```

Listing 5. A More Substantial haresources File

```
deimos \
        IPaddr::192.168.12.1 \
        Filesystem::/dev/etherd/e1.0::/opt/storage::xfs \
        killnfsd \
        nfs-common \
        nfs-kernel-server
```

Troubleshooting

Now that your cluster is all configured, start it with:

```
/etc/init.d/heartbeat start
```

Things might work perfectly or not at all. Fortunately, with logging enabled, troubleshooting is easy, because Heartbeat outputs informative log messages. Heartbeat even will let you know when a previous log message is not something you have to worry about. When bringing a new cluster on-line, I usually open an SSH terminal to each cluster member and tail the messages file like so:

```
tail -f /var/log/messages
```

Then, in separate terminals, I start up Heartbeat. If there are any problems, it is usually pretty easy to spot them.

Heartbeat also comes with very good documentation. Whenever I run into problems, this documentation has been invaluable. On my system, it is located under the /usr/share/doc/ directory.

Conclusion

I've barely scratched the surface of Heartbeat's capabilities here. Fortunately, a lot of resources exist to help you learn about Heartbeat's more-advanced features. These include active/passive and active/active clusters with N number of nodes, DRBD, the Cluster Resource Manager and more. Now that your feet are wet, hopefully you won't be quite as intimidated as I was when I first started learning about Heartbeat. Be careful though, or you might end up like me and want to cluster everything.■

Daniel Bartholomew has been using computers since the early 1980s when his parents purchased an Apple IIe. After stints on Mac and Windows machines, he discovered Linux in 1996 and has been using various distributions ever since. He lives with his wife and children in North Carolina.

Resources

The High-Availability Linux Project: **www.linux-ha.org**

Heartbeat Home Page: **www.linux-ha.org/Heartbeat**

Getting Started with Heartbeat Version 2:
www.linux-ha.org/GettingStartedV2

An Introductory Heartbeat Screencast:
linux-ha.org/Education/Newbie/InstallHeartbeatScreencast

The Linux-HA Mailing List:
lists.linux-ha.org/mailman/listinfo/linux-ha

Figure 1. HOBBIT OpenVMS Cluster Hardware

Figure 2. Linux E-Mail Server Blades and SAN

Building a Scalable High-Availability E-Mail System with Active Directory and More

A large-scale implementation of a scalable Linux e-mail system with Active Directory.

Jack Chongjie Xue

In early 2006, Marshall University laid out a plan to migrate HOBBIT (Figure 1), an HP OpenVMS cluster handling university-wide e-mail services. Plagued with increasing spam attacks, this cluster experienced severe performance degradation. Although our employee e-mail store was moved to Microsoft Exchange in recent years, e-mail routing, mailing list and student e-mail store (including IMAP and POP3 services) were still served by OpenVMS with about 30,000 active users. HOBBIT's e-mail software, PMDF, provided a rather limited feature set while charging a high licensing fee. A major bottleneck was discovered on its external disk storage system: the dated storage technology resulted in a limited disk I/O throughput (40MB/second at maximal) in an e-mail system doing intensive I/O operations.

To resolve the existing e-mail performance issues, we conducted brainstorming sessions, requirements analysis, product comparison and test-lab prototyping. We then came up with the design of our new e-mail system: it is named MUMAIL (Figure 2) and uses standard open-source software (Postfix, Cyrus-IMAP and MySQL) installed on Red Hat Enterprise Linux. The core system consists of front-end e-mail hub and back-end e-mail store. The front-end e-mail hub uses two Dell blade servers running Postfix on Linux. Network load balancing is configured to distribute load between them. The back-end e-mail store consists of two additional blade servers running a Cyrus-IMAP aggregation setup. Each back-end node is then attached to a different storage group on the EMC Storage Area Network (SAN). A fifth blade server is designated as a master node to store centralized user e-mail settings. Furthermore, we use LDAP and Kerberos to integrate the e-mail user identities with Windows Active Directory (AD).

Figure 3 illustrates our new e-mail system architecture and the subsystem interactions with existing services, which include Webmail, AD and SMTP gateway. The block diagrams highlighted in red are the components to be studied in detail.

Related Solutions

Before we zoom further into our new e-mail system, I want to mention some of the existing Linux/UNIX e-mail solutions in higher-education environments. First, the HEC Montréal e-mail system discussed in a *Linux Journal* article (see Resources) influenced our design, which is based on Cyrus-IMAP and Postfix. Second, we looked into Cambridge University's solution. It uses custom IMAP proxy front-end servers and multiple pairs of Cyrus-IMAP mail store servers replicating data to each other. Furthermore, Carnegie Mellon University (CMU), which originally developed Cyrus-IMAP, uses Sendmail as the front-end mail exchanger and a Cyrus-IMAP Murder Aggregator setup on the back end. Columbia University moved its e-mail system to a Cyrus-IMAP-based solution in 2006, and the University of Indiana moved to Cyrus back in 2005. Cyrus and Postfix also are used by Stanford University.

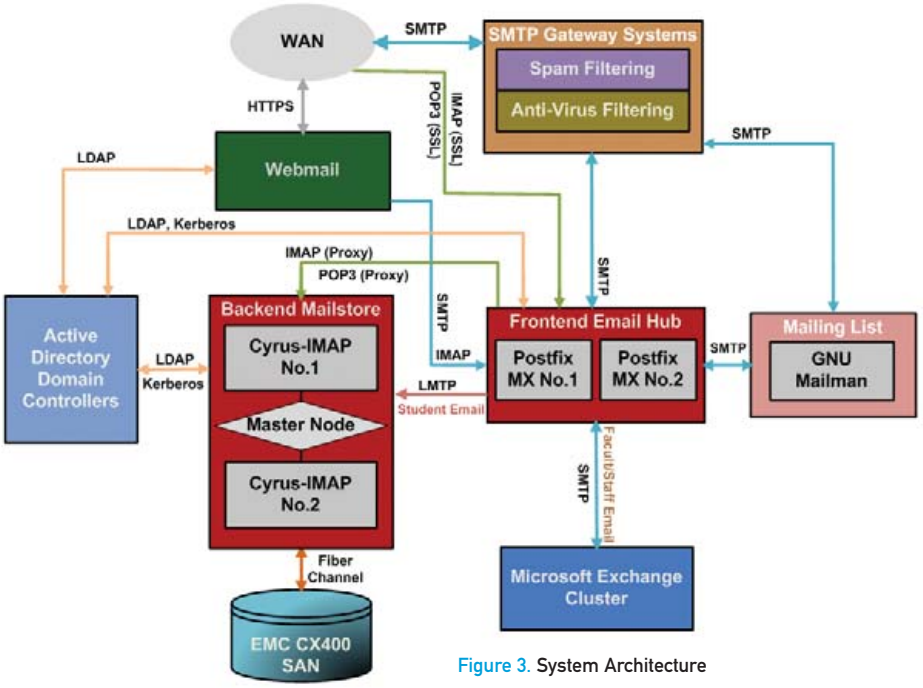


Figure 3. System Architecture

Although the designs of these related solutions are different, most of them use a cluster-based approach that separates mail transport/delivery from the mail store. Multiple front-end MTA-MDA (Mail Transport Agent and Mail Delivery Agent) servers are set up to deliver mail to the back-end mail store, which then saves messages either in a filesystem (for example, Maildir) or a database. Most of the solutions use Cyrus-IMAP (on UNIX or Linux) as their mail store server.

Detailed Design

Some distinctive differences set our design apart from the existing solutions:

1. Instead of using a separate directory service (such as OpenLDAP) for user authentication, our design integrates user identities with Windows Active Directory (AD).

2. Rather than using an LDAP server to store user e-mail routing settings, we designed a relational database to store these settings.

3. In the mail store setup, instead of using an active-passive high-availability cluster setup, like the HEC approach or the Cyrus replication approach developed at Cambridge, we deployed the Cyrus-Murder Aggregator. Unlike the CMU Cyrus Aggregator server allocation, which uses separate MTA server nodes, we consolidate both MTA and Cyrus Proxy functions to run on our front-end mail hub nodes.

We designed an e-mail user database (running MySQL on the Master node) to serve as a centralized data store for information including e-mail accounts, user e-mail routing, group aliases and mailing lists. Web-based user interfaces were developed using PHP to allow users to make changes to their settings in the database. Automated scripts running on the front-end nodes will query the database for user settings and build Postfix maps to apply these settings.

A Postfix server can be thought of as routers (not for IP packets but for e-mail). For each e-mail message, Postfix looks at the destination (envelope recipient) and the source (envelope sender) and then chooses how to route the e-mail message closer to its destination. Lookup tables called Maps (such as Transport, Virtual, Canonical and Alias Maps) are used to find the next-hop e-mail delivery location or apply e-mail address re-rewrites.

A background job is running on each of the front-end e-mail hub nodes to "pull" the e-mail settings (delivery location, e-mail alias and group alias information) stored in the e-mail user database to the Postfix maps (aliases, virtual, canonical and transport). Written in Perl, the program is configured to run periodically as a crond job.

Our design principle of the new e-mail system is to scale out from a single, monolithic architecture to multiple nodes sharing the same processing load. In a large e-mail environment, scaling out the front-end MTA system is considerably easier compared with scaling out the back-end mail store. As the front-end nodes are essentially data-less, using DNS or IP-based load balancing on multiple front-end servers is a typical practice. However, the same technique cannot be applied to design the back-end mail store where the user data resides. Without clustering, shared storage or additional software components (such as a proxy server), multiple mail store servers cannot share the same IMAP/POP3 process load under a unified service namespace. Because of this, using a single mail store server tends to be an obvious solution. However, one node usually implies elevated server hardware expenses when more powerful server hardware needs to be purchased to accommodate the ever-increasing system load. The price of a mid-range server with four CPUs is usually much higher than the total price

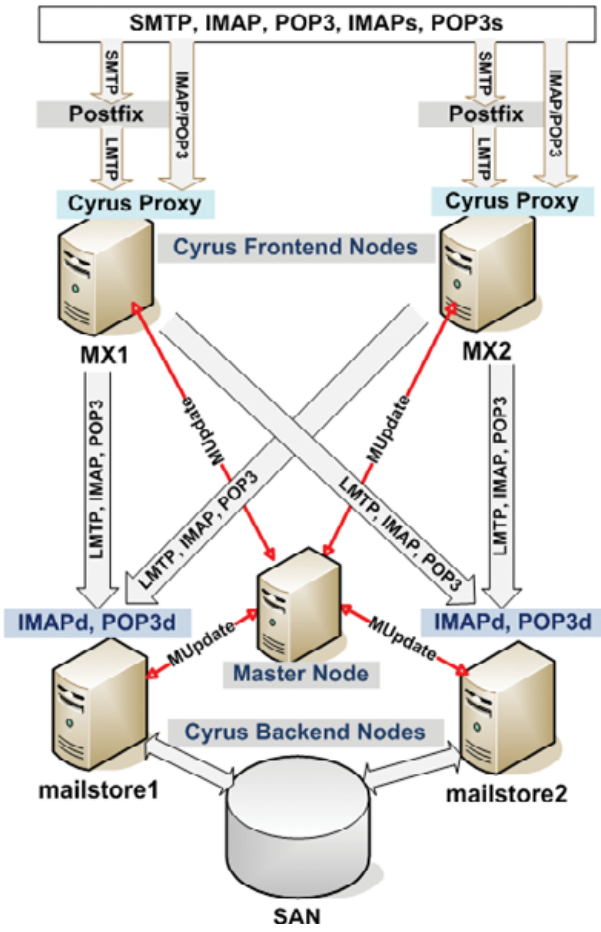


Figure 4. Cyrus-IMAP Aggregation Setup

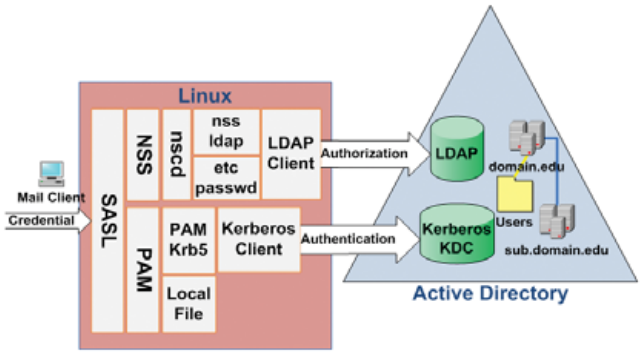


Figure 5. Linux Authentication and Authorization Against AD

of three or more entry-class servers. Furthermore, a single-node architecture reduces system scalability and creates a single point of failure.

The Cyrus-IMAP package is proven to be robust and suitable in large settings. It differs from other Maildir or mbox IMAP servers in that it is intended to run as a "sealed" mailbox server—the Cyrus mailbox database is stored in parts of the filesystem that are private to the Cyrus-IMAP system. More important, a multiple server setup using Cyrus Murder aggregation is supported. It scales out the system's load by using multiple front-end IMAP proxies to direct IMAP/POP3 traffic to multiple back-end mail store nodes. Although we found other ways to scale out Cyrus-IMAP—for example, Cambridge University's pair-wise replication approach, mentioned in the Related Solutions section of this article, or using a clustered filesystem to share IMAP storage partitions between multiple servers with products like Red Hat's Global File System (GFS)—compared with the aggregation approach, these solutions either are too customized to support (the Cambridge approach) or involve extra cost (GFS is sold separately by Red Hat, Inc.).

So, the Cyrus-IMAP Aggregation approach was adopted. Figure 4 illustrates the setup: two Cyrus back-end servers were set up, and each handles half the user population. Two Postfix MTA front-end nodes are designated to serve the proxy functions. When e-mail clients connect through SMTP/IMAP/POP3 to the front-end servers, the Cyrus Proxy service will communicate with the Cyrus Master node using the MUPDATE protocol, so that it gets the information about which Cyrus back-end node stores e-mail for the current client. Furthermore, the back-end Cyrus nodes will notify the Master node about the mailbox changes (creating, deleting and renaming mailboxes or IMAP folders) in order to keep the master updated with the most current mailbox location information. The Master node replicates these changes to the front-end proxy nodes, which direct the incoming IMAP/POP3/LMTP traffic. The MUPDATE protocol is used to transmit mailbox location changes.

Although it is not a fully redundant solution (the Master node is still a single point of failure), and half our users will suffer a usage outage if either one of the back-end nodes is down, the aggregator setup divides the IMAP processing load across multiple servers with each taking 50% of the load. As a result of this division of labor, the new mail store system is now scalable to multiple servers and is capable of handling a growing user population and increasing disk usage. More back-end Cyrus nodes can join with the aggregator to scale up the system.

Integration with Active Directory

One of the requirements of our new e-mail system is to integrate user identities with the university directory service. Because Microsoft Active Directory services have been made a standard within our centralized campus IT environment, Cyrus (IMAP/POP3) and Postfix (SMTP) are architected to obtain user authentication/authorization from AD. After the integration, all e-mail user credentials can be managed from AD. Most directory services are constructed based on LDAP. AD uses LDAP for authorization, and it has its own Kerberos implementation for authentication. The goal of an integrated AD authentication is to allow the Linux e-mail servers to use AD to verify user credentials. The technology used to support the AD integration scheme is based mainly on the Kerberos and LDAP support, which come with native Linux components, as shown in Figure 5.

Here is how it works. First, we use AD Kerberos to authenticate Linux clients. Pluggable Authentication Module (PAM) is configured to get the user credentials and pass them to the pam_krb5 library, which is then used to authenticate users using the Linux Kerberos client connection to the Key Distribution Center (KDC) on Active Directory. This practice eliminates the need for authentication administration on the Linux side. However, with only the Kerberos integration, Linux has to store authorization data in the local /etc/passwd file. To avoid managing a separate user

authorization list, LDAP is used to retrieve user authorization information from AD. The idea is to let authorization requests processed by Name Service Switch (NSS) first. NSS allows the replacement of many UNIX/Linux configuration files (such as /etc/passwd, /etc/group and /etc/hosts) with a centralized database or databases, and the mechanisms used to access those databases are configurable. NSS then uses the Name Service Caching Dæmon (NSCD) to improve query performance. (NSCD is a dæmon that provides a cache for the most common name service requests.) This can be very important when used against a large AD user container. Finally, NSS_LDAP is configured to serve as an LDAP client to connect to Active Directory to retrieve the authorization data from the AD users container. (NSS_LDAP, developed by PADL, is a set of C library extensions that allow LDAP directory servers to be used as a primary source of aliases, ethers, groups, hosts, networks, protocol, users, RPCs, services and shadow passwords.) Now, with authorization and authentication completely integrated with AD using both LDAP and Kerberos, no local user credentials need to be maintained.

In order to support LDAP authorization integration with Linux, Windows Server 2003 Release 2 (R2), which includes support for RFC 2307, is installed on each of the AD domain controllers. R2 introduces new LDAP attributes used to store UNIX or Linux user and group information. Without an extended AD LDAP schema, like the one used by R2, the Linux automatic authorization integration with AD is not possible. It is also important to mention that the SASL Authentication layer shown in Figure 3 is using Cyrus-SASL, which is distributed as a standard package by Carnegie Mellon University. The actual setup uses PAM for authenticating IMAP/POP3 users. It requires the use of a special Cyrus dæmon, saslauthd, which the SASL mechanism uses to communicate via a Linux-named socket.

Conclusion

Our new e-mail system is mostly based on open-source software. The incorporation of Postfix, Cyrus-IMAP and MySQL helped to fulfill most of the system requirements. From the hardware perspective, the technologies used, such as Storage Area Network (SAN), blade server and the Intel x86_64 CPUs, helped meet the requirements of fast access, system scalability and high availability. However, the use of open-source software and new hardware technologies may introduce new management overhead. Although all the open-source software packages used on the new system are mature products, compared with commercial software, they typically lack a GUI for system management. Their configuration and customization are completely based on a set of plain-text configuration files. Initially, this may present a learning curve, as the syntax of these configuration files must be studied. But, once the learning curve is passed, future management easily can be automated, as scripts can be written to manage the configuration parameters and store them in a centralized location. On the hardware side, complex settings also may imply complex network and server management settings, which also may introduce overhead during system management. However, the benefits of using the technologies discussed outweigh the complexities and learning curves involved. It is easy to overcome the drawbacks through proper design, configuration management and system automation.

At the time of this writing, our new Linux e-mail system (MUMAIL) has been running in production for ten months. The entire system has been running in a stable state with minimal downtime throughout this period. All user e-mail messages originally on HOBBIT were moved successfully to MUMAIL in a three-day migration window with automated and non-disruptive migration processes. Users now experience significantly faster IMAP/POP3 access speed. Their e-mail storage quota is raised from 20MB to 200MB, and there is potential to increase the quota to a higher number (1GB). With the installation of gateway-level spam/virus firewalls as well as increased hardware speed, no e-mail backlog has been experienced on MUMAIL during recent spam/virus outbreaks. With an Active Directory integrated user authentication setup, user passwords or other sensitive information are no longer stored on the e-mail system. This reduces user confusion and account administration overhead and increases network security. Mail store backup speed is improved significantly with faster disk access in the SAN environment. Finally, the new system has provided a hardware and software environment that supports future growth with the adoption of a scalable design. More server nodes—both front end and back end—and storage can be added when system usage grows in the future.■

Jack Chongjie Xue holds a Masters' degree in Information Systems from Marshall University, where he did Linux and Windows systems administration work. His job duties now include developing Business Intelligence applications and working on data mining projects at Marshall.

Resources

"Migration of Alcator C-Mod Computer Infrastructure to Linux" by T. W. Fredian, M. Greenwald and J. A. Stillerman: **www.psfc.mit.edu/~g/papers/fed04.pdf**

"HEC Montréal: Deployment of a Large-Scale Mail Installation" by Ludovic Marcotte: **www.linuxjournal.com/article/9323**

Cyrus-IMAP Aggregation: **cyrusimap.web.cmu.edu/ag.html**

"Scaling up Cambridge University's E-Mail Service" by David Carter and Tony Finch: **www-uxsup.csx.cam.ac.uk/~fanf2/hermes/doc/talks/2004-02-ukuug/paper.html**

CMU's Cyrus-IMAP Configuration: **cyrusimap.web.cmu.edu/configuration.html**

Columbia's Cyrus-IMAP Move: **www.columbia.edu/cu/news/05/12/cyrus.html**

Indiana's Cyrus-IMAP Information: **uitspress.iu.edu/040505_cyrus.html**

Stanford's E-Mail System Discussion: **www.stanford.edu/dept/its/vision/email.html**

Windows Security and Directory Services for UNIX Guide: **www.microsoft.com/downloads/details.aspx?familyid=144f7b82-65cf-4105-b60c-44515299797d&displaylang=en**

"Toward an Automated Vulnerability Comparison of Open-Source IMAP Servers" by Chaos Golubitsky: **www.usenix.org/events/lisa05/tech/golubitsky/golubitsky.pdf**

DISTRIBUTED COMPILING with distcc

You don't need a cluster to get cluster-like performance out of your compiler.

One of the most frustrating aspects of open-source development is all the time spent waiting for code to compile. Right now, compiling KDE's basic modules and libraries on a single machine takes me around three hours, and that's just to get a desktop. Even with a core 2 duo, it's a lot of time to sit around and wait.

With another pair of core duo machines at my disposal, I'd love to be able to use all of their processing power combined. Enter distcc.

distcc is a program that allows one to distribute the load of compiling across multiple machines over the network. It's essentially a front end to GCC that works for C, C++, Objective C and Objective C++ code. It doesn't require a large cluster of compile hosts to be useful—significant compile time decreases can be seen by merely adding one other similarly powered machine. It's a very powerful tool in a workplace or university environment where you have a lot of similar workstations at your disposal, but one of my favourite uses of distcc is to be able to do development work on my laptop from the comfort of the café downstairs and push all the compiles up over wireless to my more powerful desktop PC upstairs. Not only does it get done more quickly, but also the laptop stays cooler.

It's not necessary to use the same distribution on each system, but it's strongly recommended that you use the same version of GCC. Unless you have set up cross-compilers, it's also required that you use the same CPU architecture and the same operating system. For example, Linux (using ELF binaries) and some BSDs (using a.out) are not, by default, able to compile for each other. Code can miscompile in many creative and frustrating ways if the compilers are mismatched.

JES HALL

Installation

The latest version of distcc, at the time of this writing, is 2.18.3. There are packages for most major distributions, or you can download the tarball and compile it. It follows the usual automake procedure of `./configure; make; make install`; see the README and INSTALL files for details.

distcc needs to be called in place of the compiler. You simply can export CC=distcc for the compilers you want to replace with it, but on a development workstation, I prefer something a little more permanent. I like to create symlinks in ~/bin, and set it to be at the front of my PATH variable. Then, distcc always is called. This approach used to work around some bugs in the version of ld that was used in building KDE, and it is considered to have the widest compatibility (see the distcc man page for more information):

```
mkdir ~/bin for i in cc c++ gcc g++; do ln -s  `which distcc` ~/bin/$i; done
```

If ~/bin is not already at the beginning of your path, add it to your shellrc file:

```
export PATH=~/bin:$PATH
setenv PATH  ~/bin:$PATH
```

for bourne- and C-compatible shells, respectively.

Client Configuration

Each client needs to run the distcc dæmon and needs to allow connections from the master host on the distcc port (3632). The dæmon can be started manually at boot time by adding it to rc.local or bootmisc.sh (depending on the distribution) or even from an inetd. If distccd is started as an unprivileged user account, it will retain ownership by that UID. If it is started as root, it will attempt to change to the distcc or nobody user. If you want to start the dæmon as root (perhaps from an init script) and change to a user that is not distcc or nobody, the option -user allows you to select which user the dæmon should run as:

```
distccd -user jes -allow 192.168.80.0/24
```

In this example, I also use the -allow option. This accepts a hostmask in common CIDR notation and restricts distcc access to the hosts specified. Here, I restrict access only to servers on the particular subnet I'm using on my home network—machines with addresses in the 192.168.80.1–192.168.80.254 range. If you are particularly security-conscious, you could restrict it to a single address (192.168.80.5) or any range of addresses supported by this notation. I like to leave it pretty loose, because I often change which host is the master depending on what I'm compiling and when.

Compiling

Back on the master system on which you plan to run your compiles, you need to let distcc know where the rest of your cluster is. There are two ways of achieving this. You can add the hostnames or IP addresses of your cluster to the file ~/.distcc/hosts, or you can export the variable DISTCC_HOSTS delimited by whitespace. These names need to resolve—either add the names you want to use to /etc/hosts, or use the IP addresses of the hosts if you don't have internal DNS:

```
192.168.80.128 192.168.80.129 localhost
```

The order of the hosts is extremely important. distcc is unable to determine which hosts are more powerful or under less load and simply distributes the compile jobs in order. For jobs that can't be run in parallel, such as configure tests, this means the first host in the list will bear the brunt of the compiling. If you have machines of varying power, it can make a large difference in compile time to put the most powerful machines first and the least powerful machine last on the list of hosts.

Depending on the power of the computer running distcc, you may not want to include localhost in the list of hosts at all. Localhost has to do all of the preprocessing—a deliberate design choice that means you don't need to ensure that you have the same set of libraries and header files on each machine—and also all of the linking, which is often hugely processor-intensive on a large compile. There is also a certain small amount of processing overhead in managing shipping the files around the network to the other compilers. As a rule of thumb, the distcc documentation recommends that for three to four hosts, localhost probably should be placed last on the list, and for greater than five hosts, it should be excluded altogether.

Now that you have your cluster configured, compiling is very similar to how you would have done it without distcc. The only real difference is that when issuing the make command, you need to specify multiple jobs, so that the other machines in the cluster have some work to do. As a general guide, the number of jobs should be approximately twice the number of CPUs available. So, for a setup with three single-core machines, you would use `make -j6`. For three dual-core machines, you would use `make -j 12`. If you have removed localhost from your list of hosts, don't include its CPU or CPUs in this reckoning.

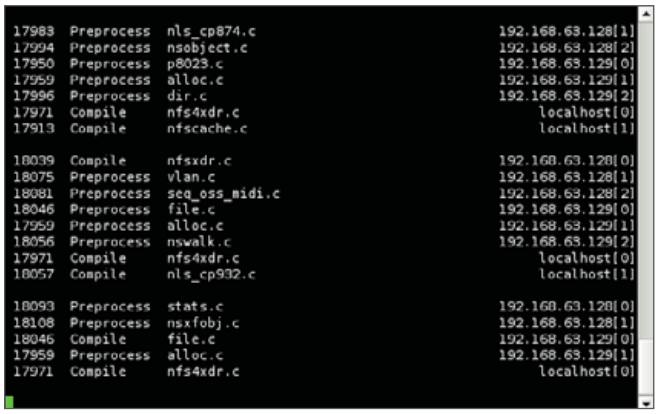


Figure 1. distccmon-text Monitoring a Compile

distcc includes two monitoring tools that can be used to watch the progress of compile jobs. The console-based distccmon-text is particularly excellent if your master host is being accessed via SSH. As the user the compile job is running as, execute the command `distccmon-text $s`, where $s is the number of seconds at which you would like it to refresh. For example, the following:

```
distccmon-text 5
```

updates your monitor every five seconds with compile job information.

The graphical distccmon-gnome is distributed as part of distcc if you compile from source, but it may be a separate package depending on your distribution. It provides similar information in a graphical display that allows you to see at a glance which hosts are being heavily utilised and whether jobs are being distributed properly. It often takes

It doesn't require a large cluster of compile hosts to be useful—significant compile time decreases can be seen by merely adding one other similarly powered machine.

a few tries to get the order of hosts at the most optimal—tools like distccmon-gnome make it easier to see whether machines are being under- or over-utilised and require moving in the build order.

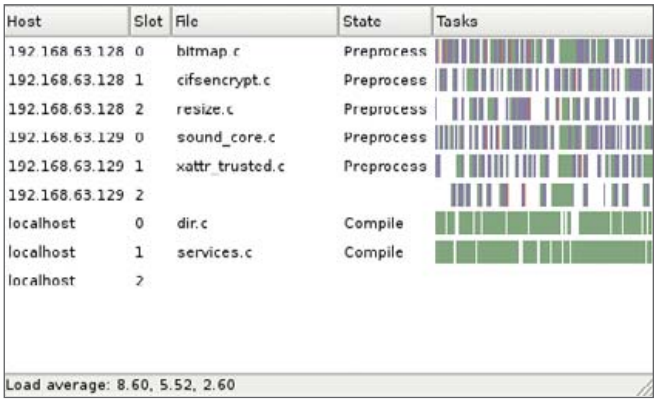


Figure 2. Graphical distcc Monitoring

Security

distcc relies on the network being trusted. Anyone who is able to connect to the machine on the distcc port can run arbitrary commands on that host as the distcc user. It is vitally important that distccd processes are not run as root but are run as the distcc or nobody user. It's also extremely important to think carefully about your -allow statements and ensure that they are locked down appropriately for your network.

In an environment, such as a home or small workplace network, where you're security firewalled off from the outside world and people can be made accountable for not being good neighbours on the network, distcc is *secure enough*. It's extremely unlikely that anyone could or would exploit your distccd hosts if they're not running the dæmon as root and your allow statements limit the connecting machines appropriately.

There is also the issue that those on the network can see your distcc traffic—source code and object files on the wire for anyone to reach out and examine. Again, on a trusted network, this is unlikely to be a problem, but there are situations in which you would not want this to happen, or could not allow this to happen, depending on what code you are compiling and under what terms.

On a more hostile network, such as a large university campus or a workplace where you know there is a problem with security, these could become serious issues.

For these situations, distcc can be run over SSH. This ensures both authentication and signing on each end, and it also ensures that the code is encrypted in transit. SSH is typically around 25% slower due to SSH encryption overhead. The configuration is very similar, but it requires the use of ssh-keys. Either passphraseless keys or an ssh-agent must be used, as you will be unable to supply a password for distcc to use. For SSH connections, distccd must be installed on the clients, but it must not be listening for connections—the dæmons will be started over SSH when needed.

First, create an SSH key using `ssh-keygen -t dsa`, then add it to the target user's ~/.ssh/authorized_keys on your distcc hosts. It's recommended always to set a passphrase on an SSH key for security.

In this example, I'm using my own user account on all of the hosts and a simple bash loop to distribute the key quickly:

```
for i in 192.168.80.120 192.168.80.100; do cat ~/.ssh/id_dsa.pub
➥| ssh jes@$i 'cat - >> ~/.ssh/authorized_keys'; done
```

To let distcc know that it needs to connect to the hosts under SSH, modify either the ~/.distcc/hosts file or $DISTCC_HOSTS variable. To instruct distcc to use SSH, simply add an @ to the beginning of the hostname. If you need to use a different user name on any of the hosts, you can specify it as user@host:

```
localhost @192.168.80.100 @192.168.80.120
```

Because I'm using a key with a passphrase, I also need to start my SSH agent with `ssh-add` and enter my passphrase. For those unfamiliar with ssh-agent, it's a tool that ships with OpenSSH that facilitates needing to enter the passphrase for your key only once a session, retaining it in memory.

Now that we've set up SSH keys and told distcc to use a secure connection, the procedure is the same as before—simply `make -jn`.

Other Options

This method of modifying the hostname with the options you want distcc to honour can be used for more than specifying the connection type. For example, the option /limit can be used to override the default number of jobs that will be sent to the distccd servers. The original limit is four jobs per host except localhost, which is sent only two. This could be increased for servers with more than two CPUs.

Another option is to use lzo compression for either TCP or SSH connections. This increases CPU overhead, but it may be worthwhile on slow networks. Combining these two options would be done with:

```
localhost 192.168.80.100/6,lzo
```

This option increases the jobs sent to 192.168.80.100 to six, and it enables use of lzo compression. These options are parsed in a specific order, so some study of the man page is recommended if you intend to use them. A full list of options with examples can be found on the distcc man page.

The flexibility of distcc covers far more than explained here. One popular configuration is to use it with ccache, a compiler cache. distcc also can be used with crossdev to cross-compile for different architectures in a distributed fashion. Now, your old SPARC workstation can get in on the act, or your G5 Mac-turned-Linux box can join the party. These are topics for future articles though; for now, I'm going to go play with my freshly compiled desktop environment.■

Jes Hall is a UNIX systems consultant and KDE developer from New Zealand. She's passionate about helping open-source software bring life-changing information and tools to those who would otherwise not have them.

Hear Yourself Think Again!

WhisperStation™

Cool... Fast... Silent!

For 64-bit HPC, Gaming and Graphic Design Applications

Originally designed for a group of power hungry, demanding engineers in the automotive industry, WhisperStation™ incorporates two dual core AMD Opteron™ or Intel® EM64T™ processors, ultra-quiet fans and power supplies, plus internal sound-proofing that produce a powerful, but silent, computational platform. The WhisperStation™ comes standard with 2 GB high speed memory, an NVIDIA e-GeForce or Quadro PCI Express graphics adapter, and 20" LCD display. It can be configured to your exact hardware specification with any Linux distribution. RAID is also available. WhisperStation™ will also make a system administrator very happy, when used as a master node for a Microway cluster! Visit www.microway.com for more technical information.

Experience the "Sound of Silence".
Call our technical sales team at 508-746-7341 and design your personalized WhisperStation™ today.

AMD 64 Opteron™ intel inside XEON

Microway®
Technology you can count on℠

Picking the RapidMind

RapidMind has a mind to make advances in the use of multicore programming rapidly available.

NICHOLAS PETRELEY

Writing applications to support multiple CPU cores is not an easy task, and in some cases, it is even harder if you want to take a huge existing application and adapt it for multiple cores. So I figured the real breakthrough is likely to be years away. It seems as if RapidMind has a solution for this problem that doesn't require a massive overhaul of an existing application, and its solution is already available.

We invited RapidMind's President and CEO Ray DePaul and Founder and Chief Scientist Michael McCool to talk about RapidMind's approach to exploiting the power of multicore systems.

We deemed it important to look at RapidMind, because it seems as if we're finally entering the age of parallel processing on the desktop as chip manufacturers bump up against the practical limits of Moore's Law. Everything from graphics cards to PlayStation 3 consoles exploit parallel processing these days. I have an Intel quad-core processor in my workstation. Although I'm happy with it, I find that the only time I truly appreciate having this multicore chip is when I run multiple applications simultaneously or run multiple processes, such as with the command `make -j 5`. If anything, single-threaded applications run slower on this chip than on the single-core CPU I used to run, because each core in the Intel chip is significantly slower (2GHz vs. 3GHz).

So how does RapidMind bridge the gap between existing software and the changing computational model?

LJ: Could you give us a brief description of RapidMind, and the problem it is designed to solve?

DePaul: RapidMind is a multicore software platform that allows software organizations to leverage the performance of multicore processors and accelerators to gain a real competitive advantage in their industry. With RapidMind, you can develop parallel applications with minimal impact on your development lifecycle, costs and timelines. And, we allow you to accomplish this without the need for multithreading. You leverage existing skills, existing compilers and IDEs and take advantage of all key multicore architectures without constantly porting your application.

LJ: So is it accurate to say RapidMind is actually a library of common C/C++ operations, where the exploitation of multiple cores is largely transparent to the programmer?

McCool: RapidMind is much more than a simple library of "canned functions". In fact, it is possible to use the API to the RapidMind platform to specify an arbitrary computation, and for that computation to execute in parallel with a very high level of performance. We provide a sophisticated multicore software platform that can leverage many levels of parallelization, but at the same time allows developers to express their own computations in a very familiar, single-threaded way.

LJ: How much, if anything, does the programmer need to know about parallel processing programming techniques in order to use RapidMind?

McCool: We believe that developers are the application experts and should have some involvement in moving their applications into the parallel world. The key is to let developers leverage what they already know, rather than force them down an unfamiliar and frustrating path. RapidMind is built upon concepts already familiar to all developers: arrays and functions. It is not necessary for a developer to work directly with threads, vectorization, cores or synchronization. Fundamentally, a developer can apply functions to arrays, and this automatically invokes parallel execution. A RapidMind-enabled program is a single-threaded sequence of parallel operations and is much easier to understand, code and test than the multithreaded model of parallel programming.

LJ: Can you give us a simple code example (the includes and declaration statements that would start a typical program)?

McCool: First, you include the platform header file and optionally activate the RapidMind namespace:

```
#include <rapidmind/platform.hpp> using namespace rapidmind;
```

Next, you can declare variables using RapidMind types for numbers and arrays:

```
Value1f f; Array<2,Value3f> a, b;
```

The Value1f type is basically equivalent to a float, and the Array types are used to manage large collections of data. These can be declared anywhere you would normally declare C++ variables: as members of classes or as local or global variables.

A Program object is the RapidMind representation of a function and is created by enclosing a sequence of operations on RapidMind types between RM_BEGIN and RM_END. The operations will then be stored in the Program object. For example, suppose we want to add a value f, represented using a global variable, to every element of an array. We would create a program object prog as follows:

```
Program prog = RM_BEGIN {
  In<Value1f> c; Out<Value1f> d;
   d = c + f;
} RM_END;
```

Note that although the program may run on a co-processor, we can just refer to external values like f in the same way we would from a function definition. It is not necessary to write any other code to set up the communication between the host

RapidMind's Founder and Chief Scientist Michael McCool (left) and President and CEO Ray DePaul (right)

processor and any co-processors.

To apply this operation to array a and put the result in array b, invoking a parallel computation, we just use the program object like a function:

```
b = prog(a);
```

Of course, in real applications, program objects can contain a large number of operations, and a sequence of program objects and collective operations on arrays (such as scatter, gather and reduce) would be used.

LJ: How do you avoid the common pitfalls of parallel processing, such as deadlocks or other synchronization issues?

McCool: The semantics of the RapidMind interface does not involve explicit locking or synchronization by the developer. The platform itself automatically takes care of these issues when necessary at a lower level in the runtime platform. The developer cannot specify programs that deadlock or that have race conditions, in the same way that a Java developer cannot specify programs that have memory leaks.

LJ: I see Hewlett-Packard software ran 32.2 times faster after the software was adapted to use RapidMind. How long did it take to modify the software to use RapidMind?

McCool: Our collaboration with HP was a great test of our plat-

form. Roughly the same amount of time was taken to RapidMind-enable the application as was taken by HP to tune its single-core baseline version. The tuning by HP sped up its version by a factor of 4, whereas RapidMind running on an NVIDIA 7900 GPU outperformed that by a factor of more than 32. More recently, we have run the same code on an NVIDIA 8800 GPU and sped it up by an additional factor of 5, and we also have run the RapidMind version on our multicore CPU quad-core product and achieved a speedup of 8 over HP's version.

So the benefit to the software organization is quite startling. For the same effort, you can use RapidMind not only to get significantly higher performance on the same multicore processors you're already targeting, but you can leverage the additional performance of accelerators as well. The RapidMind version also will scale automatically to future processors with more cores.

"For the same effort, you can use RapidMind not only to get significantly higher performance on the same multicore processors you're already targeting, but you can leverage the additional performance of accelerators as well."

LJ: Is the speed increase in the HP software typical or "best case"? What software is most likely to see speed increases? Database server software? Complex queries on data warehousing? Spam filtering? Web browsers? Something else?

McCool: We have seen large speedups on a wide range of applications, including database operations, image and video processing, financial modeling, pattern matching and analysis, many different kinds of scientific computation—the list goes on and on. The RapidMind platform supports a general-purpose programming model and can be applied to any kind of computation. The HP test was compute-bound, and it could take advantage of the high compute performance of GPUs. However, in memory-bound applications, we have also seen a significant benefit, over an order of magnitude, from running the application on RapidMind. RapidMind not only manages parallel execution, it also manages data flow and so can also directly address the memory bottleneck. As a software platform company, we are constantly surprised by the variety of applications that developers are RapidMind-enabling. Prior to the launch of our v2.0 product in May 2007, we had more than 1,000 developers from many different industries in our Beta program. The problem is industry-wide, and we have developed a platform that has very broad applicability.

LJ: Shouldn't this kind of adaptation to multiple cores take place in something more fundamental like the GNU C Library? Is it only a matter of time before such libraries catch up?

McCool: Simply parallelizing the standard library functions would not have the same benefit, because they do not, individually, do enough work. RapidMind programs, in contrast, can do an arbitrary amount of user-specified parallel computation.

Although RapidMind looks like a library to the developer, it's important to realize that most of the work is done by the runtime platform. The challenge facing multicore developers is not one that can be solved solely with libraries. Developers need a system that efficiently takes care of the complexities of multicore: processor-specific optimization, data management, dynamic load balancing, scaling for additional cores and multiple levels of parallelization. The RapidMind platform performs all of these functions.

LJ: You support multiple platforms on different levels. For example, you can exploit the processors on NVIDIA and ATI graphics cards, the Cell processor, as well as multicore CPUs. In addition, you support both Linux and Windows, correct?

DePaul: The processor vendors are delivering some exciting and disruptive innovations. Software companies are faced with some tough choices—which vendors and which architectures should they support. By leveraging RapidMind, they get to benefit from all of the hardware innovations and deliver better products to their customers within their current development cycles and timelines. RapidMind will continue to provide portable performance across a range of both processors and operating systems. We will support future multicore and many-core processors, so applications written with RapidMind today are future-proofed and can automatically take advantage of new architectures that will likely arise, such as increases in the number of cores.

LJ: Can you tell us more about your recently demonstrated support for Intel and AMD multicore CPUs?

DePaul: It's difficult to overstate the value we bring to software companies targeting Intel and AMD multicore CPUs. For example, at SIGGRAPH in San Diego, we demonstrated a 10x performance improvement on an application running on eight CPU cores. RapidMind-enabled applications will scale to any number of cores, even across multiple processors, and will be tuned for both Intel and AMD architectures. Software organizations can now target multicore CPUs, as well as accelerators, such as ATI and NVIDIA GPUs and the Cell processor, all with the same source code.

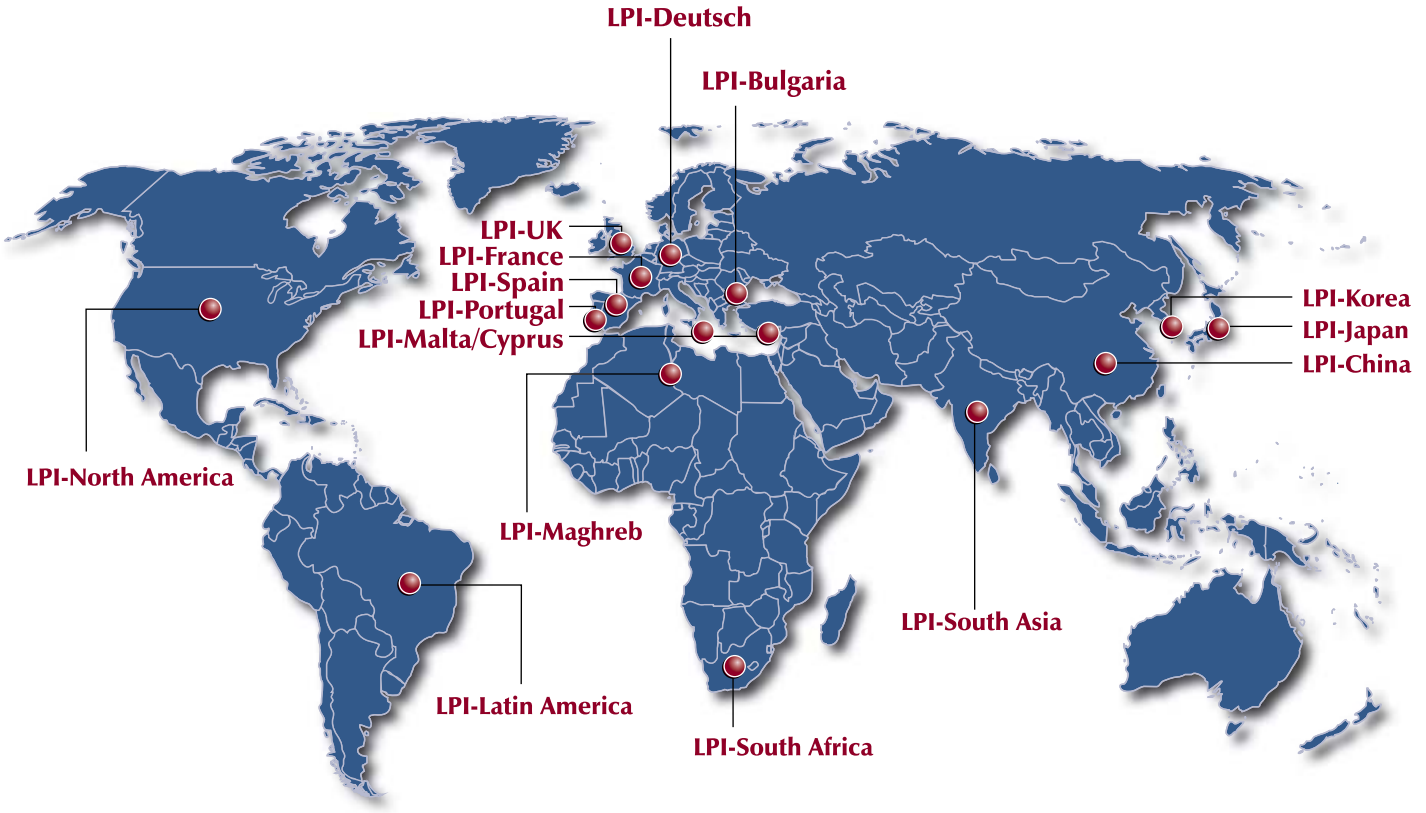
LJ: Is there anything else you'd like to tell our readers?

DePaul: It's becoming clear that software organizations' plans for multicore processors and accelerators will be one of the most important initiatives they take this year. Companies that choose to do nothing will quickly find themselves behind the performance curve. Companies that embark on large complex multithreading projects will be frustrated with the costs and timelines, and in the end, largely disappointed with the outcome. We are fortunate to be partnering with a group of software organizations that see an opportunity to deliver substantial performance improvements to their customers without a devastating impact on their software development cycles.

LJ: Thank you so much for your time! ■

Nicholas Petreley is Editor in Chief of *Linux Journal* and a former programmer, teacher, analyst and consultant who has been working with and writing about Linux for more than ten years.

Growing a World of Linux Professionals



We at the Linux Professional Institute believe the best way to spread the adoption of Linux and Open Source software is to grow a world wide supply of talented, qualified and accredited IT professionals.

We realize the importance of providing a global standard of measurement. To assist in this effort, we are launching a Regional Enablement Initiative to ensure we understand, nurture and support the needs of the enterprise, governments, educational institutions and individual contributors around the globe.

We can only achieve this through a network of local "on the ground" partner organizations. Partners who know the sector and understand the needs of the IT work force. Through this active policy of Regional Enablement we are seeking local partners and assisting them in their efforts to promote Linux and Open Source professionalism.

We encourage you to contact our new regional partners listed above.

Together we are growing a world of Linux Professionals.

Linux Professional Institute

Stable. Innovative. Growing.

High-Performance Network Programming in C

Programming techniques to get the best performance from your TCP applications.

GIRISH VENKATACHALAM

TCP/IP network programming in C on Linux is good fun. All the advanced features of the stack are at your disposal, and you can do lot of interesting things in user space without getting into kernel programming.

Performance enhancement is as much an art as it is a science. It is an iterative process, akin to an artist gingerly stroking a painting with a fine brush, looking at the work from multiple angles at different distances until satisfied with the result.

The analogy to this artistic touch is the rich set of tools that Linux provides in order to measure network throughput and performance. Based on this, programmers tweak certain parameters or sometimes even re-engineer their solutions to achieve the expected results.

I won't dwell further upon the artistic side of high-performance programming. In this article, I focus on certain generic mechanisms that are guaranteed to provide a noticeable improvement. Based on this, you should be able to make the final touch with the help of the right tools.

I deal mostly with TCP, because the kernel does the bandwidth management and flow control for us. Of course, we no longer have to worry about reliability either. If you are interested in performance and high-volume traffic, you will arrive at TCP anyway.

What Is Bandwidth?

Once we answer that question, we can ask ourselves another useful question, "How can we get the best out of the available bandwidth?"

Bandwidth, as defined by Wikipedia, is the difference between the higher and lower cutoff frequencies of a communication channel. Cutoff frequencies are determined by basic laws of physics—nothing much we can do there.

But, there is a lot we can do elsewhere. According to Claude Shannon, the practically achievable bandwidth is determined by the level of noise in the channel, the data encoding used and so on. Taking a cue from Shannon's idea, we should "encode" our data in such a way that the protocol overhead is minimal and most of the bits are used to carry useful payload data.

TCP/IP packets work in a packet-switched environment. We have to contend with other nodes on the network. There is no concept of dedicated bandwidth in the LAN environment where your product is most likely to reside. This is something we can control with a bit of programming.

Non-Blocking TCP

Here's one way to maximize throughput if the bottleneck is your local LAN (this might also be the case in certain crowded ADSL deployments). Simply use multiple TCP connections. That way, you can ensure that you get all the attention at the expense of the other nodes in the LAN. This is the secret of download accelerators. They open multiple TCP connections to FTP and HTTP servers and download a file in pieces and reassemble it at multiple offsets. This is not "playing" nicely though.

We want to be well-behaved citizens, which is where non-blocking I/O comes in. The traditional approach of blocking reads and writes on the network is very easy to program, but if you are interested in filling the pipe available to you by pumping packets, you must use non-blocking TCP sockets. Listing 1 shows a simple code fragment using non-blocking sockets for network read and write.

Note that you should use fcntl(2) instead of setsockopt(2) for setting the socket file descriptor to non-blocking mode. Use poll(2) or select(2) to figure out when the socket is ready to read or write. select(2) cannot figure out when the socket is ready to write, so watch out for this.

How does non-blocking I/O provide better throughput? The OS schedules the user process differently in the case of blocking and non-blocking I/O. When you block, the process "sleeps", which

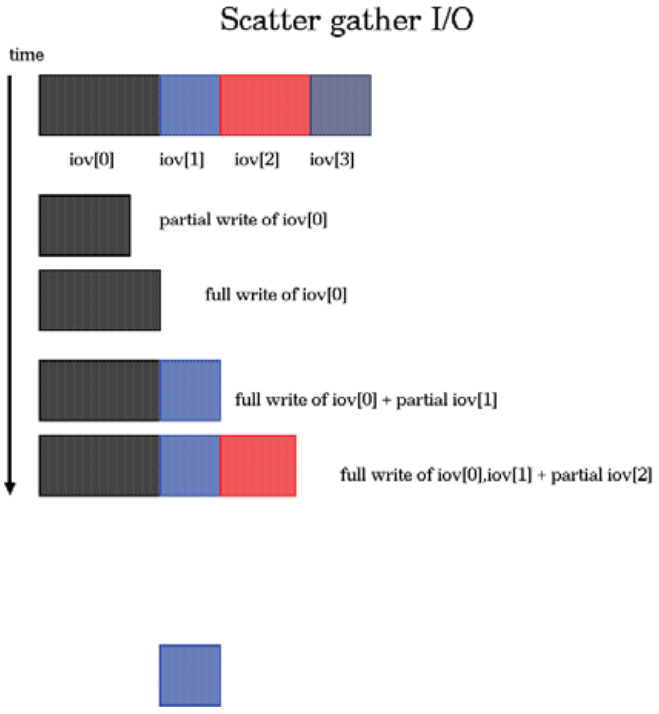


Figure 1. Possibilities in Non-Blocking Write with Scatter/Gather I/O

Listing 1. nonblock.c

```
/* set socket non blocking */
fl = fcntl(accsock, F_GETFL);
fcntl(accsock, F_SETFL, fl | O_NONBLOCK);


void
poll_wait(int fd, int events)
{
    int n;
    struct pollfd pollfds[1];
    memset((char *) &pollfds, 0, sizeof(pollfds));

    pollfds[0].fd = fd;
    pollfds[0].events = events;

    n = poll(pollfds, 1, -1);
    if (n < 0) {
        perror("poll()");
        errx(1, "Poll failed");
    }
}

size_t
readmore(int sock, char *buf, size_t n) {

        fd_set rfds;
        int ret, bytes;



        poll_wait(sock,POLLERR | POLLIN );
        bytes = readall(sock, buf, n);

        if (0 == bytes) {
                perror("Connection closed");
                errx(1, "Readmore Connection closure");
                /* NOT REACHED */
        }

        return bytes;
}

size_t
readall(int sock, char *buf, size_t n) {
        size_t pos = 0;
        ssize_t res;

        while (n > pos) {
                res = read (sock, buf + pos, n - pos);
                switch ((int)res) {
                        case -1:
                                if (errno == EINTR || errno == EAGAIN)
                                        continue;
                                return 0;
                        case 0:
                                errno = EPIPE;
                                return pos;
                        default:
                                pos += (size_t)res;
                }
        }
        return (pos);
}


size_t
writenw(int fd, char *buf, size_t n)
{
        size_t pos = 0;
        ssize_t res;
        while (n > pos) {
                poll_wait(fd, POLLOUT | POLLERR);
                res = write (fd, buf + pos, n - pos);
                switch ((int)res) {
                        case -1:
                                if (errno == EINTR || errno == EAGAIN)
                                        continue;
                                return 0;
                        case 0:
                                errno = EPIPE;
                                return pos;
                        default:
                                pos += (size_t)res;
                }
        }
        return (pos);

}
```

leads to a context switch. When you use non-blocking sockets, this problem is avoided.

Scatter/Gather I/O

The other interesting technique is scatter/gather I/O or using readv(2) and writev(2) for network and/or disk I/O.

Instead of using buffers as the unit of data transfer, an array of buffers is used instead. Each buffer can be a different length, and this is what makes it so interesting.

You can transfer large chunks of data split between multiple sources/destinations from/to the network. This could be a useful technique, depending upon your application. Listing 2 shows a code snippet to illustrate its use.

When you combine scatter/gather I/O with non-blocking sockets, things get a little complex, as shown in Figure 1. The code for tackling this hairy issue is shown in Listing 3.

A partial write of any buffer can occur, or you can get any combination of a few full writes and few partial writes. Therefore, the while loop has to take care of all such possible combinations.

mmap(2) Disk I/O

Network programming is not all about sockets, however. We still haven't solved the problem of having to use hard disks, which are mechanical devices and consequently are much slower than main memory and even the network in many, if not most, cases (especially high-performance computing environments).

You can use some other form of persistent storage, but today, none matches the huge storage capacity that hard disks offer. Currently, most applications on the Internet push several gigabytes of data, and you end up with heavy storage needs anyway.

To test disk performance, type this:

```
$ hdparm -rT /dev/sda (/dev/hda if IDE)
```

Check whether you are getting good throughput. If not, enable DMA and other safe options using this command:

```
$ hdparm -d 1 -A 1 -m 16 -u 1 -a 64 /dev/sda
```

Listing 2. uio.c

```
#include <sys/types.h>
#include <sys/uio.h>
#include <unistd.h>

size_t
writeuio(int fd, struct iovec *iov, int cnt)
{
        size_t pos = 0;
        ssize_t res;
        n = iov[0].iov_cnt;
        while (n > pos) {
                poll_wait(fd, POLLOUT | POLLERR);
                res = writev (fd, iov[0].iov_base + pos, n - pos);
                switch ((int)res) {
                        case -1:
                                if (errno == EINTR || errno == EAGAIN)
                                        continue;
                                return 0;
                        case 0:
                                errno = EPIPE;
                                return pos;
                        default:
                                pos += (size_t)res;
                }
        }
        return (pos);

}
```

We also need to be able to avoid redundant copies and other time-consuming CPU operations to squeeze the maximum bandwidth from the network. A very effective tool for achieving that is the versatile mmap(2) system call. This is a very useful technique for avoiding the copy-to-buffer cache and, hence, improves performance for network I/O. But, if you use mmap(2) with NFS, you are asking for trouble. Listing 4 shows a code snippet that illustrates the use of mmap(2) for both reading and writing files.

Socket Options and sendfile(2)

TCP sockets under Linux come with a rich set of options with which you can manipulate the functioning of the OS TCP/IP stack. A few options are important for performance, such as the TCP send and receive buffer sizes:

```
sndsize = 16384;
setsockopt(socket, SOL_SOCKET, SO_SNDBUF, (char *)&sndsize,
          (int)sizeof(sndsize));
rcvsize = 16384;
setsockopt(socket, SOL_SOCKET, SO_RCVBUF, (char *)&rcvsize,
          (int)sizeof(rcvsize));
```

I am using conservative values here. Obviously, it should be much higher for Gigabit networks. These values are determined by the bandwidth delay product. Interestingly, I have never found this to be an issue, so I doubt if this would give you a performance boost. It still is worth mentioning, because the TCP window size alone can give you optimal throughput.

Other options can be set using the /proc pseudo-filesystem under Linux (including the above two), and unless your Linux distribution turns off certain options, you won't have to tweak them.

It is also a good idea to enable PMTU (Path Maximum Transmission Unit) discovery to avoid IP fragmentation. IP fragmentation can affect not just performance, but surely it's more important regarding performance than anything else. To avoid fragmentation at any cost, several HTTP servers use conservative packet sizes. Doing so is not a very good thing, as there is a corresponding increase in protocol overhead. More packets mean more headers and wasted bandwidth.

Instead of using write(2) or send(2) for transfer, you could use the sendfile(2) system call. This provides substantial savings in avoiding redundant copies, as bits are passed between the file descriptor and socket descriptor directly. Be aware that this approach is not portable across UNIX.

Advanced Techniques in Application Design

Applications should be well designed to take full advantage of network resources. First and foremost, using multiple short-lived TCP connections between the same two endpoints for sequential processing is wrong. It will work, but it will hurt performance and cause several

Listing 3. nonblockuio.c

```
writeiovall(int fd, struct iov *iov, int nvec) {

        int i, bytes;

        i = 0;
        while (i < nvec) {
                do
                {
                        rv = writev(fd, &vec[i], nvec - i);
                } while (rv == -1 &&
                                (errno == EINTR || errno == EAGAIN));

                if (rv == -1) {
                        if (errno != EINTR && errno != EAGAIN) {
                                perror("write");
                        }
                        return -1;
                }
                bytes += rv;
                /* recalculate vec to deal with partial writes */
                while (rv > 0) {
                        if (rv < vec[i].iov_len) {
                                vec[i].iov_base = (char *)
vec[i].iov_base + rv;
                                vec[i].iov_len -= rv;
                                rv = 0;
                        }
                        else {
                                rv -= vec[i].iov_len;
                                ++i;
                        }
                }
        }

        /* We should get here only after we write out everything */

        return 0;

}
```

other headaches as well. Most notably, the TCP TIME_WAIT state has a timeout of twice the maximum segment lifetime. Because the round-trip time varies widely in busy networks and networks with high latency, oftentimes this value will be inaccurate. There are other problems too, but if you design your application well, with proper protocol headers and PDU boundaries, there never should be a need to use different TCP connections.

Take the case of SSH, for instance. How many different TCP streams are multiplexed with just one connection? Take a cue from it.

You don't have to work in lockstep between the client and the server. Simply because the protocols and algorithms are visualized in a fixed sequence does not imply that the implementation should follow suit.

Listing 4. mmap.c

```
/*****************************************
 * mmap(2) file write                    *
 *                                       *
 *****************************************/
          caddr_t *mm = NULL;

          fd = open (filename, O_RDWR | O_TRUNC | O_CREAT, 0644);

         if(-1 == fd)
         errx(1, "File write");
         /* NOT REACHED */

         /* If you don't do this, mmapping will never
          * work for writing to files
          * If you don't know file size in advance as is
          * often the case with data streaming from the
          * network, you can use a large value here. Once you
          * write out the whole file, you can shrink it
          * to the correct size by calling ftruncate
          * again
          */
         ret = ftruncate(ctx->fd,filelen);

         mm = mmap(NULL, header->filelen, PROT_READ | PROT_WRITE,
                         MAP_SHARED, ctx->fd, 0);
         if (NULL == mm)
         errx(1, "mmap() problem");
         memcpy(mm + off, buf, len);
         off += len;
         /* Please don't forget to free mmap(2)ed memory!  */
         munmap(mm, filelen);
         close(fd);
```

```
/*****************************************
 * mmap(2) file read                     *
 *                                       *
 *****************************************/
fd = open(filename, O_RDONLY, 0);
if ( -1 == fd)
errx(1, " File read err");
/* NOT REACHED */

fstat(fd, &statbf);
filelen = statbf.st_size;

mm = mmap(NULL, filelen, PROT_READ, MAP_SHARED, fd, 0);

if (NULL == mm)
errx(1, "mmap() error");
/* NOT REACHED */

/* Now onward you can straightaway
 * do a memory copy of the mm pointer as it
 * will dish out file data to you
 */


bufptr = mm + off;
/* You can straightaway copy mmapped memory into the
   network buffer for sending */
memcpy(pkt.buf + filenameoff, bufptr, bytes);

/* Please don't forget to free mmap(2)ed memory!  */
munmap(mm, filelen);
close(fd);
```

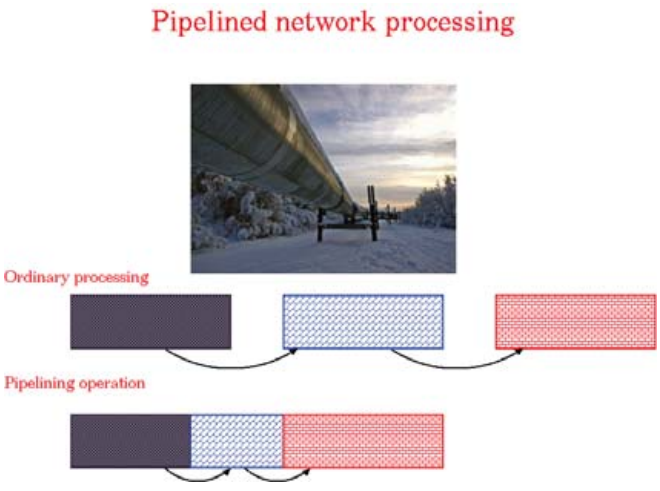


Figure 2. Pipelining

You can make excellent use of available bandwidth by doing things in parallel—by not waiting for processing to complete before reading the next packet off the network. Figure 2 illustrates what I mean.

Pipelining is a powerful technique employed in CPUs to speed up the FETCH-DECODE-EXECUTE cycle. Here, we use the same technique for network processing.

Obviously, your wire protocol should have the least overhead and should work without relying much on future input. By keeping the state machine fairly self-contained and isolated, you can process efficiently.

Avoiding redundant protocol headers or fields that are mostly empty or unused can save you precious bandwidth for carrying real data payloads. Header fields should be aligned at 32-bit boundaries and so should the C structures that represent them.

If your application already is in production and you want to enhance its performance, try some of the above techniques. It shouldn't be too much trouble to attack the problem of re-engineering an application if you take it one step at a time. And remember, never trust any theory—not even this article. Test everything for yourself. If your testing does not report improved performance, don't do it. Also,

make sure your test cases take care of LAN, WAN and, if necessary, satellite and wireless environments.

A Few Words on TCP

TCP has been a field of intense research for decades. It's an extremely complex protocol with a heavy responsibility on the Internet. We often forget that TCP is what holds the Internet together without collapse due to congestion. IP connects networks together, but TCP ensures that routers are not overloaded and that packets do not get lost.

Consequently, the impact of TCP on performance is higher than any other protocol today. It is no wonder that top-notch researchers have written several papers on the topic.

The Internet is anything but homogeneous. There is every possible physical layer of technology on which TCP/IP works today. But, TCP is not designed for working well through wireless networks. Even a high-latency satellite link questions some of TCP's assumptions on window size and round-trip time measurement.

And, TCP is not without its share of defects. The congestion control algorithms, such as slow start, congestion avoidance, fast retransmit, fast recovery and so on, sometimes fail. When this happens, it hurts your performance. Normally, three duplicate ACK packets are sufficient for triggering congestion control mechanisms. No matter what you do, these mechanisms can drastically decrease performance, especially if you have a very high-speed network.

But, all else being equal, the above techniques are few of the most useful methods for achieving good performance for your applications.

Conclusion

Gunning for very high performance is not something to be taken lightly. It's dependent on heuristics and empirical data as well as proven techniques. As I mentioned previously, it is an art best perfected by practice, and it's also an iterative process. However, once you get a feel for how things work, it will be smooth sailing. The moment you build a stable base for a fast client/server interaction like this, building powerful P2P frameworks on top is no great hassle.■

Girish Venkatachalam is an open-source hacker deeply interested in UNIX. In his free time, he likes to cook vegetarian dishes and actually eat them. He can be contacted at girish1729@gmail.com.

Resources

Polipo User Manual:
www.pps.jussieu.fr/~jch/software/polipo/manual

TCP Tuning and Network Troubleshooting: **www.onlamp.com/pub/a/onlamp/2005/11/17/tcp_tuning.html**

Wikipedia's Definition of Bandwidth:
en.wikipedia.org/wiki/Bandwidth

Advanced Networking Techniques: **beej.us/guide/bgnet/output/html/multipage/advanced.html**

TCP and Congestion Control Slides:
www.nishida.org/soi1/mgp00001.html

Multiple Associations with Stream Control Transmission Protocol

The elegant way SCTP handles multiple streams makes it ideal for things like chat clients.

JAN NEWMARCH

In two previous articles [in the September and October 2007 issues of *LJ*], I looked at the basics of SCTP, how you can use SCTP as a replacement for TCP and how you can use SCTP to process multiple streams within a single association. In this final article, I look at how a single endpoint deals with multiple associations (connections to other endpoints). First though, I explain how SCTP can give extra information about what is going on through events.

Events

The SCTP stack can generate events when "interesting" things happen. By default, all event generation is turned off except for data events. In the last article, I discussed the SCTP call sctp_rcvmsg(). By default, this just returns the data read. But, I also wanted to find out on which stream the data came, and for this I had to turn on the data_io_event so the SCTP stack would fill in the sctp_sndrcvinfo structure, which has the sinfo_stream field. Events are listed in the sctp_event_subscribe structure:

```
struct sctp_event_subscribe {
    uint8_t sctp_data_io_event;
    uint8_t sctp_association_event;
    uint8_t sctp_address_event;
    uint8_t sctp_send_failure_event;
    uint8_t sctp_peer_error_event;
    uint8_t sctp_shutdown_event;
    uint8_t sctp_partial_delivery_event;
    uint8_t sctp_adaptation_layer_event;
    uint8_t sctp_authentication_event;
};
```

An application sets fields to one for events it is interested in and zero for the others. It then makes a call to setsockopt() with SCTP_EVENTS. For example:

```
memset(&event, 0, sizeof(event));
event.sctp_data_io_event = 1;
event.sctp_association_event = 1;
setsockopt(fd, IPPROTO_SCTP, SCTP_EVENTS,
           &event, sizeof(event));
```

Events are delivered inline along with "ordinary" data whenever a read (using sctp_recvmsg or similar) is done. If the application turns on events, reads will contain a mixture of events and data. The application then will need to examine each read to see whether it is an event or data to be processed. This is quite straightforward. If the flags field in the sctp_recvmsg() call has the MSG_NOTIFICATION bit set, the read message contains an event; otherwise, it contains data as before. Pseudo-code for this is:

```
nread = sctp_rcvmsg(..., msg, ..., &flags);
if (flags & MSG_NOTIFICATION)
    handle_event(msg);
else
    handle_data(msg, nread);
```

Events can be used to tell the following: if a new association has started or if an old one has terminated; if a peer has changed state by, say, one of the interfaces becoming unavailable or a new interface becoming available; if a send has failed, a remote error has occurred or a remote peer has shut down; if partial delivery has failed; and if authentication information is available.

If an event is received in the event buffer, first its type must be found, and then the buffer can be cast to a suitable type for that event. For example, the code to handle a shutdown event is:

```
void handle_event(void *buf) {
    union sctp_notification *notification;
    struct sn_header *head;

    notification = buf;
    switch(notification->sn_header.sn_type) {
    case SCTP_SHUTDOWN_EVENT: {
        struct sctp_shutdown_event *shut;
        shut = (struct sctp_shutdown_event *) buf;
        printf("Shutdown on assoc id %d\n",
                shut->sse_assoc_id);
        break;
    }
    default:
        printf("Unhandled event type %d\n",
               notification->sn_header.sn_type);
    }
}
```

Closing an Association

A socket can support multiple associations. If you close a socket, it

closes all of the associations! It is sometimes desirable to close only a single association but not the socket, so that the socket can continue to be used for the other associations.

SCTP can abort an association or close it gracefully. Graceful shutdown will ensure that any queued messages are delivered properly before shutdown, while abort does not do this. Either of these are signaled by setting the sinfo_flags in the sctp_sndrcvinfo structure to the appropriate value. A graceful shutdown is signaled by setting the shutdown flag and writing a message (with no data):

```
sinfo.sinfo_flags = SCTP_EOF; sctp_send(..., &sinfo, ...);
```

The reader then will be sent an sctp_shutdown_event if it has that event type enabled. The code to handle such an event was shown above. This can be done only on one-to-many sockets though. For one-to-one sockets, you are limited to using close().

Getting the Association ID

Many of the calls that deal with associations take an association ID as a parameter. Whereas in TCP, a connection effectively is represented by the pair of source and destination endpoint IP addresses, in SCTP, the source and destination can both be multihomed, so they will be represented by the set of source and the set of destination addresses. For one-to-many sockets, the source addresses may be shared by many associations, so I need the destination addresses to identify an association properly. For a single association, these destination addresses all belong to a single endpoint computer. The SCTP variation on getsockopt()—that is, sctp_opt_info()—is used to find an association from an address. The reason I cannot simply use getsockopt() is that I need to pass in a socket address, and the return value includes the association value. This in/out semantics is not supported by all implementations of getsockopt(). The code is:

```
sctp_assoc_t get_associd(int sockfd, struct sockaddr *sa, socklen_t salen) {
    struct sctp_paddrinfo sp;
    int sz;

    sz = sizeof(struct sctp_paddrinfo);
    bzero(&sp, sz);
    memcpy(&sp.spinfo_address, sa, salen);
    if (sctp_opt_info(sockfd, 0, SCTP_GET_PEER_ADDR_INFO, &sp, &sz) == -1)
        perror("get assoc");
    return (sp.spinfo_assoc_id);
}
```

Note that *Unix Network Programming* (volume 1, 3rd ed.) by W. Richard Stevens, et al., gives different code: the specification has changed since that book was written, and the above is now the preferred way (and Stevens' code doesn't work under Linux anyway).

Multiple Associations

A server can handle multiple clients in a number of ways: a TCP server can use a single server socket that listens for clients and deals with them sequentially, or it could fork off each new client connection as a separate process or thread, or it could have many sockets and poll or select between them. A UDP server typically will keep no client state and will treat each message in its entirety as a separate entity. SCTP offers another variation, roughly halfway between TCP and UDP.

An SCTP socket can handle multiple long-lived associations to many endpoints simultaneously. It supports the "connection-oriented" semantics of TCP by maintaining an association ID for each association. On the other hand, it is like UDP in that each read usually returns a complete message from a client. SCTP applications use the TCP model by using the one-to-one sockets that I have discussed in the previous two articles. And, it uses a one-to-many model, which is more like UDP by using a one-to-many socket. When you create a socket, you specify whether it is one-to-one or one-to-many. In the first article in this series, I created a one-to-one socket by the call:

```
sockfd = socket(AF_INET, SOCK_STREAM, IPPROTO_SCTP)
```

To create a one-to-many socket, I simply change the second parameter:

```
sockfd = socket(AF_INET, SOCK_SEQPACKET, IPPROTO_SCTP)
```

A TCP server handles multiple connections simultaneously by essentially using concurrent reads. This is done by using multiple processes, threads, or by poll/select among many sockets. A UDP server typically uses a single read loop, handling each message as it arrives. An SCTP one-to-many server looks like a UDP server: it will bind a socket and listen. Then, instead of blocking on accept(), which would return a new one-to-one socket, it blocks on sctp_rcvmsg(), which returns a message from either a new or existing association. Pseudo-code for such a server is:

```
sockfd = socket(...);
bind(sockfd, ...);
listen(sockfd, ...);
while (true) {
    nread = sctp_rcvmsg(sockfd, ..., buf, ..., &info);
    assoc_id = sinfo.sinfo_assoc_id;
    stream = sinfo.sinfo_stream;
    handle_message(assoc_id, stream, buf, nread);
}
```

A client also can use the one-to-many socket model. After binding to a port (probably an ephemeral one), it can use the single socket to connect to many other endpoints and use this single socket to send messages to any of them. It even can do away with an explicit connect operation and just start sending to new endpoints (an implicit connection is done if no existing association exists).

Peeled-Off Sockets

One-to-one sockets follow the TCP model; one-to-many sockets follow the UDP model. Is it possible to have both at once? Yes, it is, to some extent. For example, you may have a server that you can talk to in two modes: ordinary user and superuser. Messages from ordinary users may be handled in UDP style, reading and just responding, while superuser connections may need to be treated differently. SCTP allows a connection on a one-to-many socket to be "peeled off" and become a one-to-one socket. This one-to-one socket may then be treated in TCP-style, while all other associations remain on the one-to-many socket.

Listing 1. chat_client.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/select.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

#include <netinet/sctp.h>

#define SIZE 1024
char buf[SIZE];
#define STDIN 0
char *msg = "hello\n";
#define ECHO_PORT 2013

int main(int argc, char *argv[]) {
        int sockfd;
        int nread, nsent;
        int flags, len;
        struct sockaddr_in serv_addr;
        struct sctp_sndrcvinfo sinfo;
        fd_set readfds;

        if (argc != 2) {
                fprintf(stderr, "usage: %s IPaddr\n", argv[0]);
                exit(1);
        }
        /* create endpoint using  SCTP */
        sockfd = socket(AF_INET, SOCK_SEQPACKET,
                        IPPROTO_SCTP);
        if (sockfd < 0) {
                perror("socket creation failed");
                exit(2); }
        /* connect to server */
        serv_addr.sin_family = AF_INET;
        serv_addr.sin_addr.s_addr = inet_addr(argv[1]);
        serv_addr.sin_port = htons(ECHO_PORT);

        if (connect(sockfd,
                    (struct sockaddr *) &serv_addr,
                    sizeof(serv_addr)) < 0) {
                perror("connect to server failed");
                exit(3);
        }
        printf("Connected\n");

        while (1) {
                /* we need to select between messages FROM the user
                   on the console and messages TO the user from the
                   socket
                */
                FD_CLR(sockfd, &readfds);
                FD_SET(sockfd, &readfds);
                FD_SET(STDIN, &readfds);
                printf("Selecting\n");
                select(sockfd+1, &readfds, NULL, NULL, NULL);

                if (FD_ISSET(STDIN, &readfds)) {
                        printf("reading from stdin\n");
                        nread = read(0, buf, SIZE);
                        if (nread <= 0 )
                                break;
                        sendto(sockfd, buf, nread, 0,
                               (struct sockaddr *) &serv_addr,
                               sizeof(serv_addr));
                } else if (FD_ISSET(sockfd, &readfds)) {
                        printf("Reading from socket\n");
                        len = sizeof(serv_addr);
                        nread = sctp_recvmsg(sockfd, buf, SIZE,
                                (struct sockaddr *) &serv_addr,
                                &len,
                                &sinfo, &flags);
                        write(1, buf, nread);
                }
        }
        close(sockfd);
        exit(0);
}
```

Lazy Person's Chat

In this section, I discuss a simple example of how to build a simple chat server using SCTP. This isn't meant to be a competitor to the many chat systems around, rather it is to show some of the features of SCTP.

A chat server must listen for messages coming from a probably transient group of clients. When a message is received from any one client, it should send the message back out to all of the other clients.

UDP could be a choice here: a server simply can wait in a read loop, waiting for messages to come in. But, to send them back out, it needs to keep a list of clients, and this is a bit more difficult. Clients will come and go, so some sort of "liveness" test is needed to keep the list up to date.

SCTP is a better choice: it can sit in a read loop too, but it also keeps a list of associations and, better, keeps that list up to date by sending heartbeat messages to the peers. The list management is handled by SCTP.

TCP also could be a choice: each client would start a new client socket on the server. The server then would need to keep a list of the client sockets and do a poll/select between them to see if anyone is sending a message. Again, SCTP is a better choice: in the one-to-many mode, it will keep only a single socket, and there is no need for a poll/select loop.

When it comes to sending messages back to all the connected clients, SCTP makes it even easier—the flag SCTP_SENDALL that can can be set in the sctp_sndrcvinfo field of sctp_send(). So a server simply

needs to read a message from any client, set the SCTP_SENDALL bit and write it back out. The SCTP stack then will send it to all live peers! There are only a few lines of code:

```
nread = sctp_recvmsg(sockfd, buf, SIZE,
                     (struct sockaddr *) &client_addr,
                     &len, &sinfo, &flags);
bzero(&sinfo, sizeof(sinfo));
sinfo.sinfo_flags |= SCTP_SENDALL;
sctp_send(sockfd, buf, nread, &sinfo, 0);
```

The SCTP_SENDALL flag has been introduced only recently into SCTP and is not in current kernels (up to 2.6.21.1), but it should make it into the 2.6.22 kernels. The full code for client and server is shown in Listings 1 (chat_client.c) and 2 (chat_server.c).

Unordered Messages

SCTP normally delivers messages within a stream in the order in which they were written. If you don't need this, you can turn off the ordering feature. This can make delivery of messages faster, as they don't have to be reassembled into the correct order.

New Protocols

I have examined in these three articles how TCP applications can be moved to SCTP and discussed the new features of SCTP. So, why isn't everyone using SCTP now? Well, there is the inertia of moving people off the TCP applications onto the SCTP versions, and that will happen only when people become fed up with the TCP versions—and that may never happen.

The place to look for SCTP is in new applications using new protocols designed to take advantage of SCTP:

- SS7 (Signaling System 7, see Wikipedia) is a standard for control signaling in the PSTN (Public Switched Telephone Network). SS7 signaling is done out of band, meaning that SS7 signaling messages are transported over a separate data connection. This represents a significant security improvement over earlier systems that used in-band signaling. SCTP basically was invented to handle protocols like SS7 over IP. SS7 uses multihoming to increase reliability and streams to avoid the TCP problem of head-of-line blocking.

- Diameter (RFC 3588, **www.rfc-editor.org/rfc/rfc3588.txt**) is an IETF protocol to supply an Authentication, Authorization and Accounting (AAA) framework for applications, such as network access or IP mobility. A good introduction is at **www.interlinknetworks.com/whitepapers/Introduction_to_Diameter.pdf**. It replaces an earlier protocol, Radius, that ran over UDP. Diameter uses TCP or SCTP for the added reliability of these transports. A Diameter server must support both TCP and SCTP; although at present, clients can choose either. SCTP is the default, and in the future, clients may be required to support SCTP. SCTP is preferred, because it can use streams to avoid the head-of-line blocking problem that exists with TCP.

- DLM (Distributed Lock Manager, **sources.redhat.com/cluster/dlm**) is a Red Hat project currently in the kernel. This can use either TCP or SCTP. SCTP has the advantage of multihome support. Although TCP presently is the default, SCTP can be used by setting a kernel build configuration flag.

Listing 2. chat_server.c

```
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

#include <netinet/sctp.h>

#define SIZE 1024
char buf[SIZE];
#define CHAT_PORT 2013

int main(int argc, char *argv[]) {
        int sockfd, client_sockfd;
        int nread, nsent, len;
        struct sockaddr_in serv_addr, client_addr;
        struct sctp_sndrcvinfo sinfo;
        int flags;
        struct sctp_event_subscribe events;
        sctp_assoc_t assoc_id;

        /* create endpoint */
        sockfd = socket(AF_INET, SOCK_SEQPACKET,
                        IPPROTO_SCTP);
        if (sockfd < 0) {
                perror(NULL);
                exit(2);
        }
        /* bind address */
        serv_addr.sin_family = AF_INET;
        serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
        serv_addr.sin_port = htons(CHAT_PORT);
        if (bind(sockfd, (struct sockaddr *) &serv_addr,
                 sizeof(serv_addr)) < 0) {
                perror(NULL);
                exit(3); }

        bzero(&events, sizeof(events));
        events.sctp_data_io_event = 1;
        if (setsockopt(sockfd, IPPROTO_SCTP,
                       SCTP_EVENTS, &events, sizeof(events))) {
                perror("set sock opt\n");
        }

        /* specify queue */

        listen(sockfd, 5);
        printf("Listening\n");

        for (;;) {
                len = sizeof(client_addr);
                nread = sctp_recvmsg(sockfd, buf, SIZE,
                                     (struct sockaddr *) &client_addr,
                                     &len,
                                     &sinfo, &flags);

                printf("Got a read of %d\n", nread);
                write(1, buf, nread);
                /* send it back out to all associations */

                bzero(&sinfo, sizeof(sinfo));
                sinfo.sinfo_flags |= SCTP_SENDALL;

                sctp_send(sockfd, buf, nread,
                          // (struct sockaddr *) &client_addr, 1,
                          &sinfo, 0);

        }
}
```

- MPI (Message Passing Interface, **www.mpi-forum.org**) is a de facto standard for communication among the processes modeling a parallel program on a distributed memory system (according to Wikipedia). It does not specify which transport protocol should be used, although TCP has been common in the past.

Humaira Kamal, in his Master's thesis, investigated using SCTP as a transport protocol and reported favourable results. He singled out the causes as being the message-based nature of SCTP and the use of streams within an association. These examples show that SCTP is being used in a variety of real-world situations to gain benefits over the TCP and UDP transports.

Conclusion

This series of articles has covered the basics of SCTP. There are many options that can control, in fine detail, the behaviour of an SCTP stack. There also is ongoing work in bringing a security model into SCTP, so that, for example, TLS can be run across SCTP. There also is work being done on different language bindings to SCTP, such as a Java language binding. SCTP will not make TCP and UDP disappear overnight, but I hope these articles have shown that it has features that can make writing many applications easier and more robust.

Of course, SCTP is not the only attempt to devise new protocols. For comparisons to other new procotols see "Survey of Transport Protocols other than Standard TCP" at **www.ogf.org/Public_Comment_Docs/Documents/May-2005/draft-ggf-dtrg-survey-1.pdf**. This shows that SCTP stacks up very well against possible alternatives, so you might want to consider it for your next networking project!■

Jan Newmarch is Honorary Senior Research Fellow at Monash University. He has been using Linux since kernel 0.98. He has written four books and many papers and given courses on many technical topics, concentrating on network programming for the last six years. His Web site is jan.newmarch.name.

LISA'07

21ST LARGE INSTALLATION SYSTEM ADMINISTRATION CONFERENCE

See what's
DEVELOPING in
SYSTEM ADMINISTRATION!

LISA '07 offers the most in-depth, practical system administration training available!

6 DAYS OF TRAINING BY EXPERTS IN THEIR FIELDS, INCLUDING:

- Tom Limoncelli on Time Management for System Administrators
- Steve VanDevender on High-Capacity Email System Design
- Æleen Frisch on Administering Linux in Production Environments
- Faisal N. Jawdat on Introduction to Ruby, Rails, and Capistrano

3-DAY TECHNICAL PROGRAM, INCLUDING:

Keynote by John Strassner, *Motorola Fellow and Vice President, Autonomic Networking and Communications, Motorola Research Labs*, on Autonomic Administration: HAL 9000 Meets Gene Roddenberry

Invited Talks by industry leaders discuss important and timely topics, including:

- Bruce Moxon, NetApp, "A Service-Oriented Data Grid: Beyond Storage Virtualization"
- Erik Nygren, Akamai Technologies, "Experiences with Scalable Network Operations at Akamai"
- Kenneth G. Brill, Uptime Institute, "The Economic Meltdown of Moore's Law"

Refereed Papers, Hit the Ground Running Track, Guru Is In Sessions, Vendor Exhibition, Workshops, BoFs, WiPs, and more!

NOVEMBER 11-16, 2007

DALLAS

Sponsored by
USENIX & [sage]

Register by October 19 and save! www.usenix.org/lisa07/lj

Roman's Law and Fast Processing with Multiple CPU Cores

Some practical methods to exploit multiple cores and find thread synchronization problems.

ROMAN SHAPOSHNIK

Computers are slow as molasses on a cold day, and they've been like that for years. The hardware industry is doing its work, but computers are not going to get any faster unless the software industry follows suit and does something about it. Processing speed is less about GHz and more about multiple cores these days, and software needs to adapt.

I will be so bold as to postulate my own law of the multicore trend for hardware: the number of cores on a chip would double every three years. It remains to be seen whether I'm going to be as accurate in this prediction as Gordon Moore happened to be, but it looks good so far. With Intel and AMD introducing quad-core systems and Sun pushing the envelope even further with its first hexadeca-core CPU named Rock, the question the software industry has to ask itself is "Are we ready to make all these execution threads do useful work for us?" My strong opinion is that we are not ready. A new paradigm is yet to be invented, and we don't really know what it should look like. What we do know, however, in the words of Herb Sutter, is "The biggest sea change in software development since the OO revolution is knocking at the door, and its name is Concurrency."

The first layer of software where hardware parallelism is going to percolate is the kernel of your operating system. And, I know of only two kernels that have what it takes to tackle the challenge: Solaris and Linux. If there's any Mac OS X—or dare I say, Windows—fanboys out there, I have two phrases for you: "256 processor systems" and "Completely Fair Scheduler". Now, having a kernel that is capable of efficiently multiplexing a large number of processes to a large number of hardware threads is only as good as your demand in actually having that large number of individual processes running in parallel. And, as much as it is an ISP or provisioning person's dream come true, on my laptop I rarely have more than four really active processes running at the same time. The real need that I have is for each individual application to be able to take advantage of the underlying hardware parallelism. And, that's how a fundamental hardware concept permeates yet another software layer—an application one. At the end of the day we, as userland software developers, have no other choice but to embrace the parallel view of the world fully. Those pesky hardware people left us no choice whatsoever.

For the rest of this article, I assume that you have at least a dual-core system running Linux kernel 2.6.x and that you have Sun Studio Express installed in /opt/sun/sunstudio and added to your PATH, as follows:

```
PATH=/opt/sun/sunstudio12/bin:$PATH
```

My goal is to explain the kind of practical steps you can take to teach that old serial code of yours a few multicore tricks.

There are three basic steps to iterate through to add parallelism gradually to your serial application:

1. Identify parallelism.
2. Express parallelism.
3. Measure and observe.

And, even though the first two steps sound like the most exciting ones, I cannot stress enough the importance of step 3. Parallel programming is difficult and riddled with unexpected performance gotchas. And, there is no other way of being certain that your code got faster than proving it with numbers. The good news is that it isn't all that difficult. If you happen to develop mostly for Intel architectures, you can use code similar to the FFMPEG's START_TIMER/STOP_TIMER for microbenchmarking:

```
START_TIMER
do_interesting_stuff();
STOP_TIMER("do_interesting_stuff_tag")
```

Additionally, there's the Sun Studio Performance analyzer for observing macro behaviour. You also can use tools like Intel's VTune or even time(1), but whatever you do, make sure that performance regression testing is as much a part of your testing routine as the regular regression testing is. You do regression testing, don't you?

Identifying parallelism in an existing application usually starts with finding spots with data parallel characteristics, task parallel characteristics and figuring out a scheduling model to tie the two together. Data parallelism usually can be found in applications working with large sets of global or static data (think audio, video and image processing, gaming engines and rendering software). Task parallelism, on the other hand, mostly is appropriate when branch-and-bound computation takes place (think Chess-solvers

when a bunch of tasks are asked to do similar calculations, but if one finds a solution, there's no need to wait for others).

Once you've identified all potential sources of the parallelism in your application, you have to decide what programming techniques to use for expressing it. For an application written in C or C++, the most commonly used one happens to be explicit parallelization with POSIX threads. This method has been around for decades, and most developers usually have some familiarity with it. On the other hand, given its inherent complexity and the fact that it no longer is the only game in town, I'm going to skip over it.

Let's look at this sample code, which happens to be a very simple routine for calculating how many prime numbers there are between 2 and N:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <omp.h>

/* pflag[v] == 1 if and only if v is a prime number   */
char *pflag;

int is_prime(int v)
{
    int i;
    int bound = floor(sqrt(v)) + 1;

    for (i = 2; i < bound; i++) {
        if (v % i == 0) {
            pflag[v] = 0;
            return 0; /* v is NOT a prime number */
        }
    }
    return 1; /* v is a prime number */
}

int main(int argc, char **argv)
{
    int i;
    int total = 0;
    int N = atoi(argv[1]);
    int primes[N];     /* array of prime numbers */
    pflag=(char*)alloca(N);

    for (i = 2; i < N; i++) {
        pflag[i] = 1;  /* all numbers are prime until... */
    }                  /* ...proven otherwise            */

    for (i = 2; i < N; i++) { /* let the testing begin! */
        if ( is_prime(i) ) {
            primes[total] = i;
            total++;
        }
}

printf("Number of prime numbers between 2 and %d: %d\n",
        N, total);
```

Advertiser Index

```
44
45  return 0;
46  }
```

Granted, the code is silly (some might even say brain-dead), but let's pretend it is a real-life application. In that case, we certainly would benefit from as much automation as possible. And, if you think about it, there's no tool better suited for helping us than a compiler—after all, it already takes care of understanding the semantics of the code in order to perform optimizations. Ideally, what we would need is a compiler that talks back to us, helping us understand the source code better and make reasonable tweaks based on that information. Here's how Sun Studio 12 lets you do that:

```
$ cc -g -fast prime.c -o prime
$ er_src prime
.....
Source loop below has tag L3
35.      for (i = 2; i < N; i++) { /* let the testing begin! */

Function is_prime inlined from source file prime.c into the code
for the following line.  1 loops inlined
Loop in function is_prime, line 14 has tag L4
36.           if ( is_prime(i) ) {
```

Finally! Your compiler actually explains to you in plain human language, what transformations it applied to the source code to make it faster (-fast). Not only that, but it also identifies and tags all key areas (such as loops) that you later can navigate and inspect for the parallelization potential. Identifying parallelism just got easier. But what about expressing parallelism? Would it be completely out of the question to delegate some of that to the compiler as well? After all, we are too lazy to use POSIX threads (besides, they are like the GOTOs of parallel programming, anyway). The good news is that with the right compiler it *is* possible. But, before we go there, let's remember the third step from our "three-step parallelization program" and establish a performance baseline:

```
$ cc -fast prime.c -o prime
$ collect ./prime 2000000
Creating experiment database test.1.er ...
Number of prime numbers between 2 and 2000000: 148933
$ er_print -statistics test.1.er
                    Execution for entire program

                                  Start Label: Total
                                    End Label: Total
                            Start Time (sec.): 0.028
                              End Time (sec.): 3.364
                              Duration (sec.): 3.336
                         Total LWP Time (sec.): 3.337
                        Average number of LWPs: 1.000
................................................
```

The -fast command-line option instructs the Sun Studio C compiler to generate the fastest possible code for the same architecture where the compilation happens. The last two commands actually run the generated executable and report runtime statistics for the function main. Now we know that whatever we do to the source code, the result shouldn't be slower than 3.336 seconds. With that in mind, let's try asking the compiler to do its best not only at identifying parallelism (-xloopinfo), but at expressing it as well (-xautopar):

```
$ cc -fast -xloopinfo -xautopar prime.c -o prime
"prime.c", line 14: not parallelized, loop has multiple exits
"prime.c", line 14: not parallelized, loop has multiple exits
                    (inlined loop)
"prime.c", line 31: PARALLELIZED, and serial version generated
"prime.c", line 35: not parallelized, unsafe dependence (total)
```

So, with only two extra command-line options, the compiler was smart enough to parallelize the loop on line 31 (-xautopar) and honest enough to explain why two other loops (lines 14 and 35) cannot be parallelized easily (-xloopinfo). That's pretty impressive, but let's see whether it got us any speedup at all:

```
$ export OMP_NUM_THREADS=4
$ collect ./prime 2000000
Creating experiment database test.2.er ...
Number of prime numbers between 2 and 2000000: 148933
$  er_print -statistics test.2.er | grep Duration
                                 Duration (sec.): 3.331
```

Good. It isn't slower (although not significantly faster either), but then again, we didn't have to do anything with the source code. The compiler did everything for us (except letting the runtime system use all the way up to four threads by setting the OMP_NUM_THREADS environment variable to four). Or did it? What about that loop on line 35? It doesn't look any more complicated than the one on line 31. Seems like the compiler is being overly conservative, and we need to step in and help it a bit. This time, let's express parallelism with OpenMP.

The formal (and boring) definition of OpenMP states that it is "an API that supports multiplatform shared-memory parallel programming in C/C++ and Fortran on all architectures, including UNIX platforms and Windows NT platforms". Personally, I'd like to think about OpenMP as a method of helping the compiler exploit data parallelism in your application when data dependencies get out of hand. In short, OpenMP is something you use when -xautopar complains. Given that, for C and C++, OpenMP is expressed through the #pragmas, it is quite safe to add these hints (although making sure that suggested parallel operations don't have concurrency problems is still your responsibility). As with any #pragma, if the compiler doesn't understand it, it'll skip over it. (At the time of this writing, the following freely available Linux compilers support OpenMP 2.5: Intel Compilers, GCC 4.2 and Sun Studio 12.)

So, how do we use OpenMP to boost the compiler's confidence in the loop on line 35? Simply add the following pragma to line 34:

```
34 #pragma omp parallel for
35 for (i = 2; i < N; i++) { /* let the testing begin! */
36      if ( is_prime(i) ) {
```

And, don't forget to add -xopenmp to the set of command-line options:

```
$ cc -fast -xloopinfo -xautopar -xopenmp prime.c -o prime
"prime.c", line 14: not parallelized, loop has multiple exits
"prime.c", line 14: not parallelized, loop has multiple exits
                    (inlined loop)
"prime.c", line 31: PARALLELIZED, and serial version generated
"prime.c", line 35: PARALLELIZED, user pragma used
```

Nice! We've got two out of three loops in our application now completely parallelized. Let's see how much faster it runs:

```
$ collect ./prime 2000000
Creating experiment database test.3.er ...
Number of prime numbers between 2 and 2000000: 146764
$ er_print -statistics test.3.er | grep Duration
                                Duration (sec.): 1.132
```

No doubt, 294% is an impressive speedup, but how come we lost some prime numbers? We now have 146,764 of them reported instead of 148,933. Maybe the compiler was right in being conservative and not parallelizing that pesky loop. Should we go back and remove our OpenMP pragma? Not yet. Even though we've just caught a bug in our parallelized version of the same application (which only goes to show how easy it is to introduce bugs and how much more important regression testing becomes when you try to parallelize anything), we still are on the right track. The question is, how do we find the actual problem?

The trouble with parallel programming is that it makes some sections of your application nondeterministic. That means now you have to deal with all sorts of problems you didn't have to deal with before, such as race conditions, and deadlock and resource allocation issues are the chief nemeses. The amount of nondeterminism introduced is, in fact, something that makes POSIX threads quite fragile in most real-life situations—so much so, that one of the key parallel computing researchers, Professor Edward A. Lee, made a pretty scary prediction in his article "The Problem with Threads":

> I conjecture that most multithreaded general-purpose applications are, in fact, so full of concurrency bugs that as multicore architectures become commonplace, these bugs will begin to show up as system failures. This scenario is bleak for computer vendors: their next generation of machines will become widely known as the ones on which many programs crash.

As you can see, OpenMP, even though it introduces significantly less nondeterminism than POSIX threads do, is still not a panacea. After all, even our simplistic usage of it was enough to introduce a bug. It seems that regardless of how we express parallelism, what we need is a tool that would help us uncover concurrency bugs.

At the end of the day we, as userland software developers, have no other choice but to embrace the parallel view of the world fully.

I know of two such tools freely available on Linux: Intel Thread Checker and Sun Studio Thread Analyzer. And, here's how you can use the latter one to combat data races (note that we need an extra compile-time command-line option -xinstrument=datarace to make thread analysis possible and that we have to ask collect for recording data race events by specifying -r on):

```
$ cc -fast -xloopinfo -xautopar -xopenmp -xinstrument=datarace
➥prime.c -o prime
$ collect -r on ./prime 2000000
Creating experiment database tha.1.er ...
```

```
Number of prime numbers between 2 and 2000000: 148933
$ er_print -races tha.1.er
No race information recorded in experiments
```

Weird. Not only did we get the correct result, but also the thread analyzer didn't seem to notice anything unusual. Is it broken? Not really. You see, what makes concurrent bugs so notoriously difficult to track down is the fact that most of them are intermittent. As with most manifestations of a nondeterministic behavior, they come and go

The trouble with parallel programming is that it makes some sections of your application nondeterministic.

depending on how applications are run, what else runs on the system and whether you use tools such as a conventional debugger. Thread analyzer reports only those problems that did actually occur. Well, if at first you don't succeed:

```
$ collect -r on ./prime 2000000
Creating experiment database tha.2.er ...
Number of prime numbers between 2 and 2000000: 114833
$ er_print -races tha.2.er
Total Races:  2 Experiment:  tha.2.er

Race #1, Vaddr: (Multiple Addresses)
   Access 1: Write, main -- MP doall from line 34
    [_$d1B34.main] + 0x00000172, line 37 in "prime.c"
   Access 2: Write, main -- MP doall from line 34
    [_$d1B34.main] + 0x00000172, line 37 in "prime.c"
Total Traces: 1

Race #2, Vaddr: 0xbffff28c
   Access 1: Write, main -- MP doall from line 34
    [_$d1B34.main] + 0x00000189, line 38 in "prime.c"
   Access 2: Write, main -- MP doall from line 34
    [_$d1B34.main] + 0x00000189, line 38 in "prime.c"
Total Traces: 1
```

Bingo! We reproduced the bug and our tool dutifully reported the actual location of where the race condition happened: lines 37 and 38. Things go wrong when two threads find prime numbers and they try to update the primes array and total variable—a textbook example of a race condition. But, it's pretty easy to fix. We have to serialize threads entering these two lines of code. Can we do that with OpenMP? Sure we can:

```
37 #pragma omp critical
38 {
39     primes[total] = i;
40     total++;
41 }
```

With that, let's see what the final speedup is going to be:

```
$ cc -fast -xloopinfo -xautopar -xopenmp prime.c -o prime
$ collect ./prime 2000000
Creating experiment database test.4.er ...
Number of prime numbers between 2 and 2000000: 148933
$ er_print -statistics test.4.er | grep Duration
                                Duration (sec.): 1.130
```

It's 2.95 times faster. Not bad for 15 minutes of work and four extra lines of OpenMP pragmas giving hints to the compiler!

A Few Loose Ends

OpenMP and -xautopar seem to work pretty well for C, but what about C++? Will they mesh well with the kind of modern C++ usage peppered with generics and template metaprogramming? The short answer is, there's no short answer. But, let's see for ourselves with the following example of modern C++ [ab]use:

```
#include <vector>
#include <iterator>
```

Resources

Moore's Law: **www.intel.com/technology/mooreslaw/index.htm**

Sun Studio Express: **developers.sun.com/sunstudio/downloads/express/index.jsp**

FFMPEG: **ffmpeg.mplayerhq.hu**

Intel VTune: **www.intel.com/cd/software/products/asmo-na/eng/239145.htm**

Intel Thread Checker: **www.intel.com/cd/software/products/asmo-na/eng/291669.htm**

TotalView Debugger: **www.totalviewtech.com/index.htm**

POSIX Threads: **www.mhpcc.edu/training/workshop2/pthreads/MAIN.html**

OpenMP: **www.openmp.org**

Effective Use of OpenMP in Games: **https://www.cmpevents.com/Sessions/GD/EffectiveUse.ppt**

Intel Thread Building Blocks: **osstbb.intel.com**

Cilk: **supertech.csail.mit.edu/cilk**

"The Problem with Threads": **www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-1.pdf**

"The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software": **gotw.ca/publications/concurrency-ddj.htm**

```
#include <algorithm>
#include <iostream>
void standard_input_sorter() {
   using namespace std;
   vector<string> v;
   copy(istream_iterator<string>(cin), istream_iterator<string>(),
        back_inserter(v));
   sort(v.begin(), v.end());
}

$ CC -c -fast -xloopinfo -xautopar -xopenmp -library=stlport4 sorter.cc
```

The above produces a pretty long list of complaints, explaining why a particular section of the STLport library cannot be parallelized. The key issue here is that certain areas of C++ are notoriously difficult to parallelize by default. Even with OpenMP, things like concurrent container access are much more trouble than they are worth. Do we have to rewrite STL? Well, seems like Intel almost did. Intel has been working on what it calls the Thread Building Blocks (TBB) C++ library, and its claim to fame is exactly that—making modern C++ parallel. Give it a try, and see if it works for you. I especially recommend it if you're interested in exploiting task parallelism. But, then again, the amount of modern C++ that TBB throws at even the simplest of examples, such as calculating Fibonacci numbers, is something that really makes me sad. Tasks as defined in the upcoming OpenMP 3.0 standard seem far less threatening.

Conclusion

There is a fundamental trend toward concurrency in hardware. Multicore systems are now making their way into laptops and desktops. Unfortunately, unless software engineers start taking these trends into account, there's very little that modern hardware can do to make individual applications run faster. Of course, parallel programming is difficult and error-prone, but with the latest tools and programming techniques, there's much more to it than merely POSIX threads. Granted, this article scratches only the surface of what's available. Hopefully, the information presented here will be enough of a tipping point for most readers to start seriously thinking about concurrency in their applications. Our high-definition camcorders demand it and so does every gamer on earth.■

Roman Shaposhnik started his career in compilers back in 1994 when he had to write a translator for the programming language he'd just invented (the language was so weird, nobody else wanted the job). His first UNIX exposure was with Slackware 3.0, and he's been hooked ever since. Currently, he works for Sun Microsystems in the Developer Products Group. He is usually found pondering the question of how to make computers faster yet not drive application developers insane. He runs a blog at blogs.sun.com/rvs and can be reached via e-mail at rvs@sun.com.

High-Performance Linux Clusters

The present and future of high-performance computing.

DAVID MORTON

Twice a year, a group of scientists in Europe and the United States release a list of the world's 500 most powerful computing systems. The Top 500 list is the most prestigious ranking of its kind, with vendors and users leveraging favorable rankings to promote their work. The most recent list, released June 2007, reconfirmed a recent trend: Linux is by far the most frequently used operating system in high-performance computing (HPC). Consider the numbers: 389 machines (or 78%), run some flavor of Linux, 64 run UNIX, two run Windows and 42 feature a mix of Linux and other operating systems.

Although such dominance suggests that Linux has had a long history in HPC, the truth is that Linux clusters began replacing UNIX systems only six years ago. The reason for such quick initial take-up is due to the fact that Linux and open systems brought commodity hardware and software into what had previously been a proprietary systems market. This change brought costs down significantly, allowing users at the high end to purchase more power at lower cost and opening the door for new users, such as traditional product designers who were not able to afford closed proprietary systems. The domination of Linux in the HPC market is so successful that market research firm IDC estimated Linux represented 65% of the total HPC market by mid-2006 (as compared to approximately 30% for UNIX), with additional growth projected. The Top 500 list confirms that growth.

Challenges and Questions

Linux is clearly the present of HPC, but is it the future? Microsoft continues to make advancements with its Windows Compute Cluster Server, has plenty of cash on hand and is clearly capable, from a business perspective, of eating up market share. In addition, despite its well-known flaws, everyone has worked with and is familiar with Windows, potentially making it a comfortable platform to new HPC users.

Complicating matters further is that, despite their well-earned market dominance, high-performance Linux clusters have, in many cases, earned a reputation for being difficult to build and manage. Widely available commodity components lead to complexity in the selection, integration and testing required when building a stable system. This complexity becomes doubly problematic when you consider that organizations invest in HPC systems in order to get the best possible performance for the applications they run. Small variations in system architecture can have a disproportionately large impact on time to production, system throughput and the price/performance ratio.

Furthermore, like any new technology, the first high-performance Linux clusters hit bumps in the road. Early systems took a very long time for vendors to build and deliver and an even longer time to put into production. Additionally, early management software made re-provisioning systems and upgrading components cumbersome. Finally, delivering HPC systems is as much about understanding the nuances of computer-aided engineering (CAE) applications as it is about understanding technical minutiae related to interconnects, processors and operating systems. Early vendors of high-performance Linux clusters did not necessarily have the expertise in computational fluid dynamics (CFD), finite element analysis (FEA) and visualization codes of proprietary systems vendors.

It is, therefore, natural for many to question whether the tremendous price advantage of Linux and open systems still outweighs all other considerations. The truth is that although Windows provides some advantages to entry-level HPC users, high-performance Linux clusters have matured. Today's Linux clusters deliver better performance at a more attractive price than ever before. Clusters are increasingly being demanded as turnkey systems, allowing faster time to production and fewer management headaches. In addition, the very nature of open source has contributed to the strength of high-performance Linux clustering. Linux clusters adapt more quickly to new technology changes, are easier to modify and optimize and benefit from a worldwide community of developers interested in tweaking and optimizing code.

The Advantages of Linux-Based HPC

The most important factor in HPC is, of course, performance. National laboratories and universities want ever-more powerful machines to solve larger problems with greater fidelity. Aerospace and automotive engineering companies want better performing systems in order to grow from running component-level jobs (such as analyzing the stress on an engine block) to conducting more complex, multi-parameter studies. Product designers in a variety of other fields want to graduate from running CAE applications on their relatively slow workstations in order to accelerate the overall design process.

Performance, therefore, cannot be separated from high-performance computing and in this area, Linux clusters excel. There are two primary reasons for this: maturity and community.

Maturity

With years of experience under their belts, vendors and architects of high-performance Linux clusters are better equipped than ever to design stable, tuned systems that deliver the desired price performance and enable customers to get the most out of their application licenses.

First-generation systems may have been difficult to manage, but the newest generation comes equipped with advanced cluster management software, greatly simplifying operations. By selecting an experienced vendor, many of today's clusters are delivered as full-featured systems as opposed to an unwieldy pile of stitched-together commodity components. As a result, users benefit from both lower acquisition costs and easy-to-use high-performance systems.

The maturity of the Linux HPC industry also contributes to a deeper understanding of the codes users rely on, as well as the hardware that goes into building a system. Certain vendors have become experts at

tuning systems and optimizing Linux to meet and overcome the challenges posed by widely used HPC applications. For example, most high-performance structures codes, such as those from ANSYS or ABAQUS, require high I/O to sustain higher rendering rates. Conversely, crash/impact codes don't require much I/O to run optimally; they are designed to run in parallel in systems where the average CPU count is 16. Linux has evolved to the point where it is now very easy for vendors to build systems that accommodate the needs of these codes—even within the same cluster.

Alliant Techsystems (ATK) is a recent example of how high-performance Linux clusters have matured. ATK is an advanced weapon and space systems company with many years of experience working with HPC systems. In 2006, faced with upgrading its aging proprietary system, the launch system's group invested, after extensive benchmarking, in a high-performance Linux cluster—finding one tuned and optimized for CFD, FEA and visualization codes. The decision reflected their understanding that Linux clusters—and vendors—had matured.

"We had heard several horror stories of organizations that moved to Linux supercomputers, only to suffer through installation times that stretched to six or eight months and beyond", said Nathan Christensen, Engineering Manager at ATK Launch Systems Group. "For instance, one of ATK's other business units experienced eight weeks of waiting and downtime to get a system into production. The Launch Systems Group wanted to avoid a similar experience."

"The system arrived application-tuned, validated and ready for production use", said Christensen. "We were able to move quickly into full production, generating our simulations and conducting our analysis within two weeks of delivery."

The system also accelerated the company's time to results, thereby enabling ATK to complete designs faster and conduct more frequent, higher-fidelity analysis. The launch system's group completes runs three to four times faster than before. In addition, on some of its key CFD and FEA applications, ATK has been able to achieve ten times the throughput performance.

Community

The greater Linux community is also an important factor in assuring that Linux-based systems deliver the greatest performance. The benefit of being open source means that users and vendors from around the world continue to develop innovations and share them with the greater community. This enables Linux-based HPC systems to adapt more quickly to new hardware and software technologies. As a result, the ability to take advantage of new processors, interconnects and applications is much greater than with proprietary systems.

Additional Benefits

High-performance Linux clusters offer a range of benefits beyond raw application performance.

First, Linux is well known for its ability to interoperate with all types of architectures and networks. Because of the investment in HPC systems, users want to make certain that their systems are as future-proof as possible. Linux provides users with an operating system that is flexible enough to accommodate virtually any future advancement. This is further amplified, of course, when the larger Linux community, working together to solve common problems, is again taken into question. In addition, a variety of tools, such as Samba, allow Linux to share file services with Windows systems, and vice versa.

Second, Linux clusters evolved without headless operations. As a result, administrative tools are able to install and manage the system as a whole, rather than as individual workstations or servers. These tools continue to get easier to use, enabling users with limited technical skills to jump quickly into HPC. To take just one example, Linux Networx recently launched its newest cluster management application, Clusterworx Advanced. This application provides system administrators with intuitive tools that greatly simplify operations and reduce administration workload.

Third, Linux-based clusters are easy to scale due, in part, to newer filesystems, such as GPFS and Lustre, which provide better scalability, but only on Linux and UNIX. Windows-based filesystems are typically tuned for file sharing and don't provide the type of performance and accessibility required when lots of compute nodes all request the same dataset at the same time.

Fourth, resource management tools, such as Altair's PBS Pro and Platform LSF, ensure that computing resources are being allocated with utilization rates exceeding 90%. Without proper resource management, systems tend to work only when the engineering team works, thereby limiting overall utilization. With mature resource management tools, such as those available for Linux-based HPC systems, jobs can be scheduled 24 hours a day, 365 days a year. Multiple jobs can be run simultaneously, as needed, thereby ensuring excess power is always put to use.

Fifth, from a stability perspective, Linux—due to its flexibility and the number of people working on refining it—is significantly more stable and scalable than other platforms. Windows, for instance, is prone to failure at moderately larger node counts and is not considered as an option at government and national laboratories.

Sixth, the nature of open source makes Linux the most convenient platform for vendors and users to work with. Standards are broadly defined and supported by a worldwide community of programmers—rather than the diminishing numbers found at the remaining proprietary vendors. As a result, there are no shortages of fully developed tools, utilities and software modifications that users and vendors can leverage in order to optimize their systems.

Conclusion

The HPC market has made its choice, and the choice is the Linux OS due to its superior performance, lower costs and its community of open-source developers and vendors. Windows may have a lot to offer entry-level users, especially those with more limited resources or goals. Likewise, UNIX still has a lot to offer for many legacy HPC applications. However, both Windows and UNIX require more work to be able to deliver the same functionality and compelling price-performance value of Linux. The HPC market is more open and competitive than it has ever been, but it is clear that Linux is still the best choice for today and the foreseeable future.■

David Morton brings 17 years' experience in supercomputing in both vendor and end-user roles to Linux Networx. Dave is responsible for leading the Linux Networx technology vision and directing the hardware, software and system engineering teams. Previously, Dave served as technical director for the Maui High Performance Computer Center where he was responsible for the definition and oversight of technology for this Department of Defense supercomputing center. He held several positions at SGI, including director of server I/O and continuation engineering and director of Origin platform engineering. Dave's experience includes eight years at Cray Research, where he was named inventor on three issued patents. Dave earned a Master's degree in Mechanical Engineering from the University of Illinois and an MBA from the University of Minnesota.

Open-Source Compositing in Blender

How to use node-based compositing in Blender.

DAN SAWYER

Linux is bursting with multimedia potential—at least, that's the impression one gets from the plethora of multimedia-oriented distributions that have surfaced in recent years. DeMuDi, Planet CCRMA, Ubuntu Studio, 64 Studio and the list continues on ad infinitum. However, for many years now, the term multimedia proved deceptive. GIMP and Inkscape and other graphics tools meant that 2-D graphics were covered, and the astounding variety of audio tools available with real-time priority meant that users' needs for recording and audio processing were met. Video tools lagged behind, because video processing is both more difficult and more prone to patent encumbrance. In the last few years, things have begun to catch up to the point where it's feasible to create films or cartoons from concept through execution using only Linux tools.

Elephants Dream, one such cartoon, was the foundation for a major breakthrough in open-source video. Financed by presales of a then-unseen cartoon, *Elephants Dream* was a strategy for raising money to advance the development of and raise awareness for the open-source 3-D suite Blender (**www.blender.org**). In order to accomplish this goal, the creators had to develop something that never had been available before: an open-source compositor.

Compositing is the art of taking multiple image sources—whether from 3-D, vector graphics, photographs, video or procedurals—and marrying them together to create a seamless, integrated image. A good compositing program provides the means to access all the mathematical functions available in the image processing universe, and a good artist needs to be able to get down into the guts of an image from time to time, below the interface, and tweak it directly with mathematical functions.

Because of Linux's continuing adoption in post houses, several high-end compositing systems, such as Shake, D2 Nuke and Eyeon Fusion, have been available for years now,

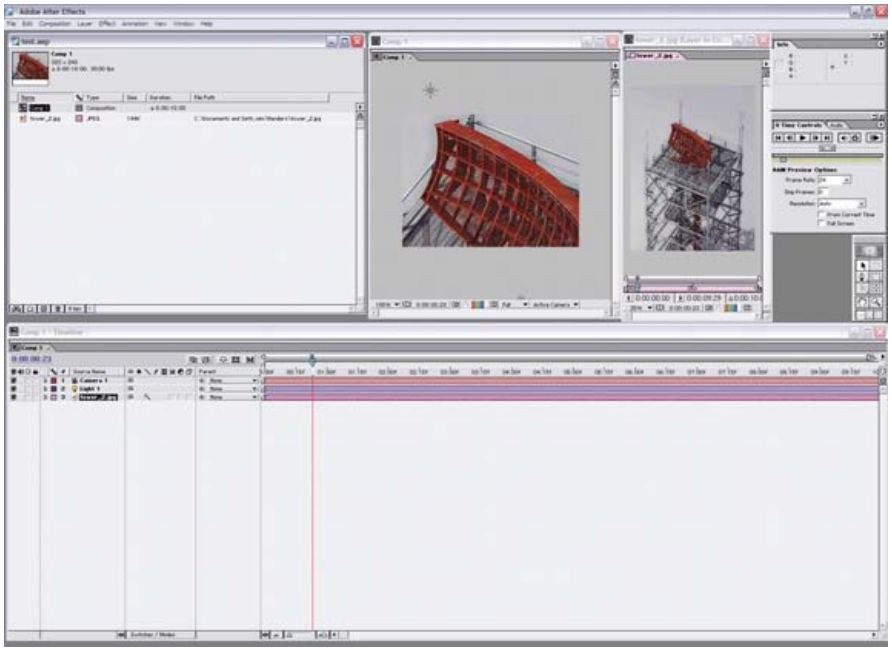


Figure 1. Adobe After Effects Interface

but the prices run up to thousands of dollars per seat with extra costs for maintenance and render-node licenses. For those with smaller budgets, Linux compositing has been perpetually out of reach, unless one has the patience to do hand scripting in ImageMagick, which is far more technical than most artists care to get and generally requires the addition of a hacker to the pipeline (another cost point), or work frame by frame in GIMP, which is laborious and not worth the effort for any but the smallest projects.

Consequently, the only budget-friendly solutions for a small studio has been Adobe's After Effects or Apple's Motion, which means adding a Windows or Mac OS machine to the pipeline. Both are very capable, useful tools that produce professional results, but neither are open source.

The two classes of compositors are built around two divergent interface paradigms, which dramatically effect work flow. The first of these paradigms is the Photoshop paradigm. It shows up most prominently in After Effects, and in a modified form in Apple Motion, and works by marrying the interface conventions of Photoshop with a basic multitrack editor interface. In this paradigm, composites are achieved using layers of images built atop each other, with each layer being operated upon by its own effects stack. The main advantages of this paradigm are the speed of work for simple and moderately complex effects and the ease of navigation for new users who already are familiar with Photoshop (Figure 1).

The second paradigm, the "node-based" paradigm, is the one that appears in the high-end professional compositors. It works by chaining together various image functions to create complex effects. Image functions are mathematical transforms applied to an image to change it in one way or another,

Figure 2. A Node-Based Interface

and they reside at the base of anything one does in GIMP or ImageMagick or in video compositing. These functions are encapsulated in the interface by nodes. A node works a bit like a guitar pedal—it accepts inputs and generates outputs, and those outputs can be routed to an infinite number of other nodes. Thus, in a node-based compositor, one uses the node chains to accomplish one's goal, and there typically are two types of nodes from which to choose. One type is the familiar, user-friendly prepackaged effects plugins, such as one would find in the Photoshop universe. The other type is a set of mathematical interfaces that allow you to build custom effects yourself. This has the disadvantage of being far more visually complex and, for some people, much harder to learn. However, for that steeper learning curve, the artist gets a much more versatile work flow, which is better suited to doing highly complex work. Node-based compositors available for Linux include: Shake (now defunct), Eyeon Fusion, D2 Nuke (formerly of Digital Domain, now owned by the Foundry) and Blender (Figure 2).

Blender itself has long had a rudimentary track-based compositing system, which has received a lot of attention since *Elephants Dream* and has become quite useful both as a video editor and a compositor. Alas, because its primary purpose is video editing, it lacks the ability to nest compositions or layer effects as complexly as can After Effects and Motion, leaving it firmly in the quick-and-dirty category for compositing work.

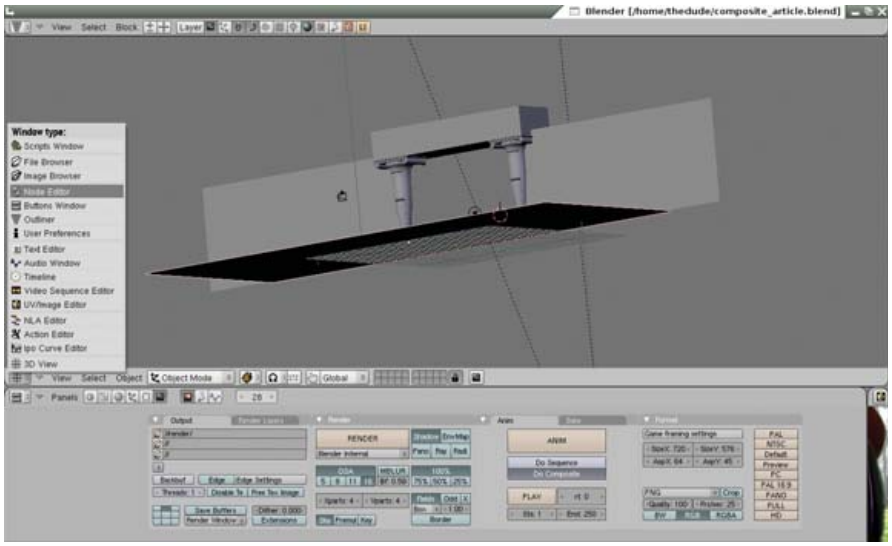


Figure 3. Finding the Nodes Editor

However, with version 2.43, Blender introduced its node-based compositor, the jewel in the crown of the *Elephants Dream* improvements. Professional-level open-source compositing has arrived, and it's integrated with an otherwise very powerful 3-D content creation suite.

To demonstrate what it can do, let's walk through a fairly simple five-layer composite.

To get to the compositor, fire up Blender and change the main 3-D window to the nodes editor (Figure 3). Toggle on the Use Nodes button. Because Blender uses nodes for material and texture creation as well as compositing, you need to depress the picture icon. By default, a render scene and a composite node will pop up. In the case of this project, one of the elements I'm using is a

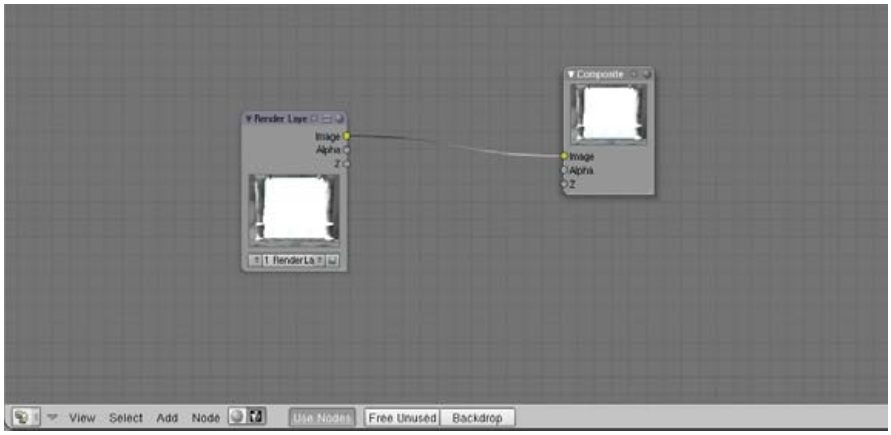


Figure 4. Basic Composite Nodes Setup

Professional-level open-source compositing has arrived, and it's integrated with an otherwise very powerful 3-D content creation suite.

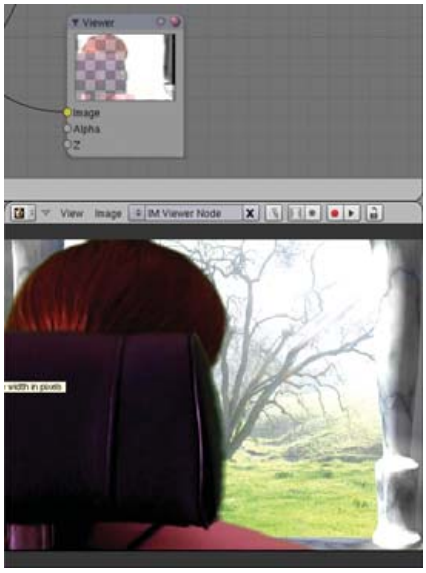


Figure 5. The Viewer Node and Window

3-D scene in Blender, so I retain this node and assign it to my primary camera (Figure 4).

Next, I split the bottom view into two windows, and in the right-most pane, pull up the image editor window, where there is a list box that allows you to choose the output nodes from the compositor window. This is how you check your progress (Figure 5).

Next, I add a few more sources. In each case, pressing space brings up a menu that allows me to add nodes. I want three additional input nodes, and from each I assign the appropriate type. For this project, I'm working with two still images (the lens flare and the background photo) and one image sequence (the greenscreen clip, rendered out from DV to an image sequence to make it easier to work with).

First, I build the nodes chain for the background image (Figure 6). The first node, moving left to right, is our source node—the photo itself. The second is a scale node, which I use to shrink my rather large digital photograph down to something approximating SD video resolution. Next, I've used an RGB curves node to blow the highlights out of the photo to help sell the illusion that our character is looking from a shaded courtyard out into a wild garden in direct sunlight.

Next, I take the pillars element, which is a rendered layer from within Blender proper, add a procedural glow layer to it, and marry the glow and the pillars to the background. To do this, I take an output from the source and run it through a curves pass to cut out all but the brightest highlights (Figure 7).

I pipe the output from the curves node into a blur node, where I do a 40-pixel x/y tent blur and then direct that into a Screen node, where the glow is composited back over the source picture of the pillars. This married image is then piped into an AlphaOver node, which pastes the pillars and glow over the top of the photo.

Now we come to the color keying. There are a number of ways to do color keying in a node-based compositor, not least among them is building your own keyer out of basic mathematical functions. Although this method is excellent and yields the best results (even on DV footage, which has some very particular problems with color keying), it requires far more ink than I have space for here. So, for the sake of brevity, I selected one of Blender's three native keying nodes, the channel key, to do the job (Figure 8).

The footage, first off, needs some prep, and I've prepared the same footage two different ways—one for pulling the matte and the other for color matching. I first ran the footage through a scaling node to correct for the 16:9 aspect ratio I shot in—as the rest of my elements are in 4:3, I'm pre-correcting the footage rather than doing it at render time. I then ran it through a translate node, which allowed me to reposition the footage to the left, so that we actually can look over the actress' shoulder rather than just staring at the back of her head. From there, I send the output into two parallel subtrees—keying and color correction.

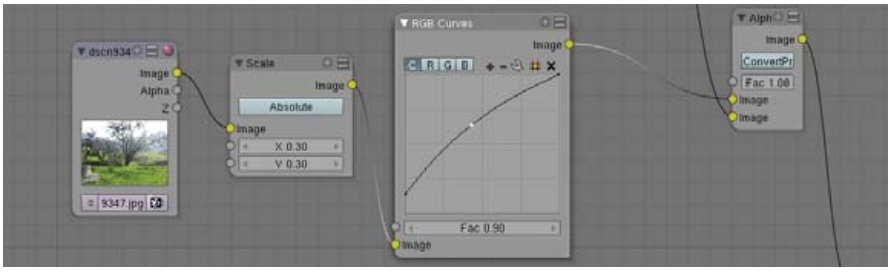


Figure 6. The Background Nodes Tree

Why an Image Sequence Instead of a Video?

Working with image sequences offers three distinct advantages over working with video. First, with image sequences it's easier to retouch problems on individual frames without reprocessing an entire clip. Second, the variety of alpha-enabled lossless formats gives the operator a far greater flexibility than do video formats, very few of which are lossless or allow embedded alpha channels. Third, video codecs are, frankly, a big pain in the neck. They vary wildly in support from one program to another; however, image formats are universal and comparatively open. Converting your video to image sequences before piping it through your compositor or motion tracker means that you're going to encounter far fewer problems with moving between various programs in your work flow.

With Blender, there is one further advantage to using image sequences, and it has to do with a shortcoming of the program. Blender does not support NTSC drop or nondrop frame encoding, both of which run at 29.97 frames per second (fps). Because it's a European program and has historically been an animation program, the closest approximation it can hit is 30fps. Though .3fps seems like a trivial difference, it's more than enough to slip your sound sync beyond the limits of intelligibility, so it's far better to work with image sequences and then re-multiplex your audio with your video in your video editing software, such as KDENLIVE or OpenMovieEditor.

The keying subtree begins with a curves node, which pushes the green in the greenscreen into a narrow band to make it easier for the keyer to latch on to. Then, the output links to the input of a channel keyer, set to pull the cleanest possible matte (which I accomplished by hooking a viewer node to the image output, so I could see what I was doing when I played with the settings). The resulting matte is then run through a blur node. Normally, when keying DV footage, I would apply a 4x2 blur to soften the matte and compensate for the edge artifacting introduced by the DV compression. However, in this case, my edges were dark, because of how I lit the original scene, and I needed some extra feathering so the brightness from the background would bleed over. The output of this blur node is then piped into the Fac input of an AlphaOver node, which marries the greenscreen footage to the rest of the image. But, I'm getting ahead of myself.

Let's back up to the other half of the keying tree. This takes an

Converting your video to image sequences before piping it through your compositor or motion tracker means that you're going to encounter far fewer problems with moving between various programs in your work flow.

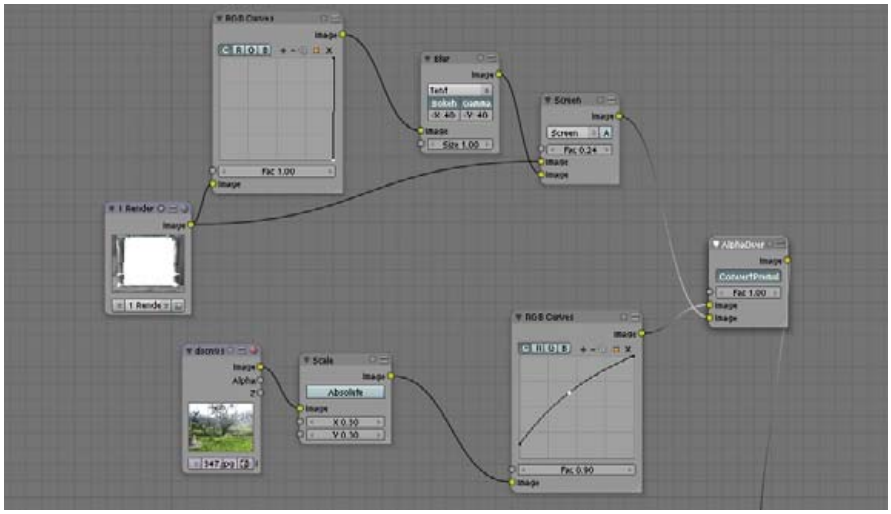


Figure 7. Pillars and Glow Pass

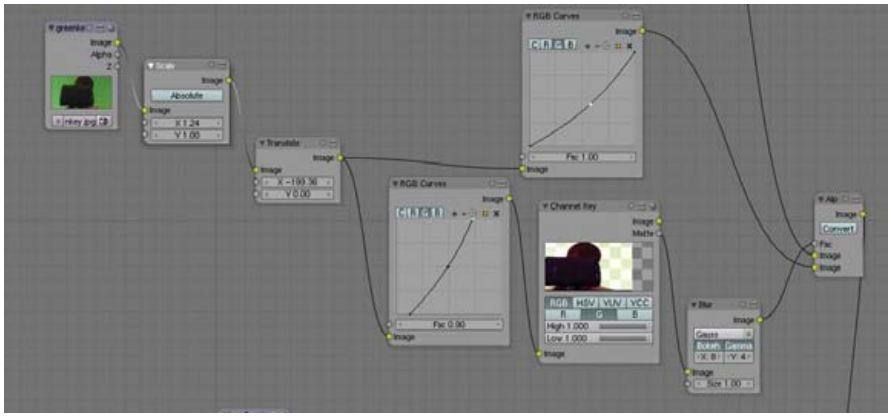


Figure 8. The Color Keying Nodes Tree

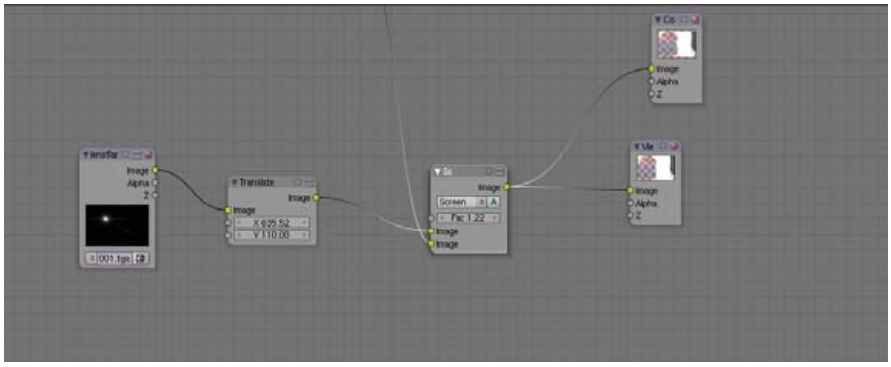


Figure 9. Layers of the Image Including Lens Flare

additional output from the translate node into a curves node, which is set to tamp down the green channel to get rid of the green spill and help sell the different lighting conditions of the foreground vs. the background. The output of this curves node is then run into the bottom input on AlphaOver. Now, to complete the marriage of foreground with background, we run an additional noodle from the AlphaOver node at the end of the background subtree into the top image input on the keyer AlphaOver node.

I could leave things here, but the shot could use a little extra touch to tie all the layers together. To accomplish this, I created a nice lens flare and brought it in to Blender. I ran it through a translate node to put it into the right spot, and from there into another screen node, which lays it over the top of the previous composite. To do this, the lens flare went into the top image input, and the previous AlphaOver node went into the bottom image input, and I messed with the Fac, so I got the right effect—just a hint of extra brightness and anamorphic smear, which helps sell the integration of the different layers (Figure 9).

Now, all that remains is to hook up the Composite node, which is what Blender draws from for its output. This can be found next to the viewer node under output in the add nodes menu, which you get to by pressing the spacebar. Once the composite node is hooked up to the output, go to the renderbuttons window at the bottom of the screen, depress the Do Composite button, and click Render, or, if it's an animation, click Anim (Figure 10). The result of your hard work appears in the render window, from where you can save it using F3 (if it's a still). Or, you can find it on your hard drive in the temp directory or another directory that you have set for output in the renderbuttons window.

Simple though this project is, it gives a good grounding in how nodes work and why they're useful. Enough access to basic

Figure 10. Click Render or Anim in the Blender Controls

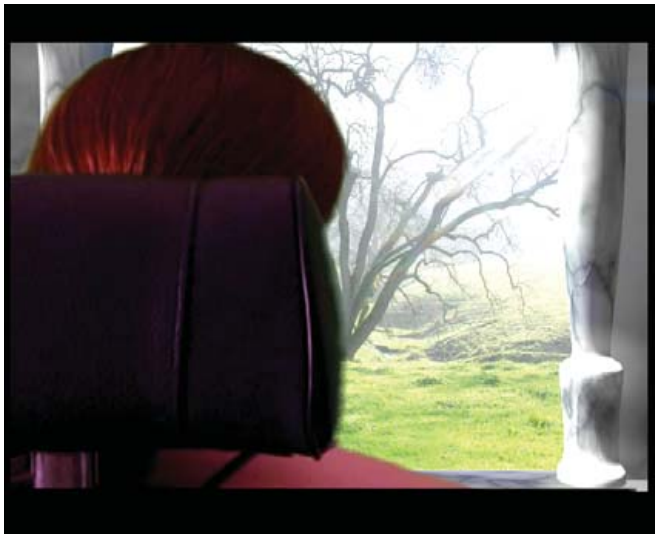


Figure 11. The Completed Project

image processing functions is included that the capabilities are very deep and extensive, and because of Blender's support for HDR formats, such as OpenEXR, and its lack of limitation on resolutions, it is squarely in the professional compositing camp, albeit at the less-sophisticated end of the spectrum (as one would expect from a brand-new project). It is advancing quickly. In future versions, more user-friendly keying tools and color tools are planned, and hopefully there also will be more direct access to the translation and garbage matting functions, which at the moment are obtuse and inconvenient. Until such tools emerge, I highly recommend that anyone wanting to use Blender as a workhorse compositor invest in a book that teaches how compositing works, both in theory and practice. The best available is *The Art and Science of Digital Compositing* (The Morgan Kaufmann Series in Computer Graphics).

Using Blender for composite work has significant advantages as well, since it's an integrated part of a 3-D content creation suite, the particle systems, fluid systems, procedural textures and all the traditional 3-D modeling and animation tools are at the compositing system's disposal, which is supremely useful for any number of highly complicated shots that normally would require using several programs in conjunction to pull off correctly.

Here's hoping the *Project Peach*, the currently in-process sequel production to *Elephants Dream*, gives us more such innovations that push the compositing system to the next plateau. Until then, there is much to explore, learn and use.

Open-source compositing has finally arrived. Enjoy!■

Dan Sawyer is the founder of ArtisticWhispers Productions (www.artisticwhispers.com), a small audio/video studio in the San Francisco Bay Area. He has been an enthusiastic advocate for free and open-source software since the late 1990s, when he founded the Blenderwars filmmaking community (www.blenderwars.com). Current projects include the independent SF feature *Hunting Kestral* and *The Sophia Project*, a fine-art photography book centering on strong women in myth.

Statement of Ownership, Management, and Circulation

1. Publication Title: *Linux Journal*
2. Publication Number: 1075-3583
3. Filing Date: September 10, 2007
4. Issue Frequency: Monthly
5. Number of Issues Published Annually: 12
6. Annual Subscription Price: $25
7. Complete Mailing Address of Known Office of Publication: PO Box 980985 Houston, TX 77098 Contact Person: Mark Irgang 713-589-3503
8. Complete Mailing Address of Headquarters of General Business Office of Publisher: PO Box 980985 Houston, TX 77098
9. Full Names and Complete Addresses of Publisher, Editor, and Managing Editor: *Publisher.* Carlie Fairchild PO Box 980985 Houston, TX 77098 *Editor.* Doc Searls PO Box 980985 Houston, TX 77098 *Managing Editor.* Jill Franklin PO Box 980985 Houston, TX 77098
10. Owner(s): Carlie Fairchild PO Box 980985 Houston, TX 77098 Joyce Searls PO Box 980985 Houston, TX 77098 Adele Soffa PO Box 980985 Houston, TX 77098
11. Known Bondholders, Mortagees, and Other Security Holders Owning or Holding 1 Percent or More of Total Amount of Bonds, Mortages, or Other Securities: None
12. Tax Status: Has not Changed During Preceding 12 Months

13. Publication Title: *Linux Journal* — 14. Issue Date: October

15. Extent and Nature of Circulation	Average No. Copies Each Issue During Preceding 12 Months	No. Copies of Single Issue Published Nearest to Filing Date
a. Total Number of Copies: (Net press run)	57,908	54,182
b. Paid and/or Requsted Circulation		
(1) Paid/Requested Outside-County Mail Subscriptions on Form 3541.	20,086	19,537
(2) Paid In-County Subscriptions Stated on Form 3541	0	0
(3) Sales Through Dealers and Carriers, Street Vendors, Counter Sales, and Other Non-USPS Paid Distribution	25,688	24,005
c. Total Paid and/or Requested Circulation	45,774	43,542
d. Free Distribution Outside the Mail		
(1) Outside-County as Stated on Form 3541	810	780
(2) In-County as Stated on Form 3541	0	0
(3) Other Classes Mailed Through the USPS	0	0
e. Free Distribution Outside the Mail	2,343	1,720
f. Total Free Distribution	3,153	2,500
g. Total Distribution	48,927	46,042
h. Copies Not Distributed	8,981	8,140
i. Total	57,908	54,182
j. Percent Paid and/or Requested Circulation	94%	95%

PS Form 3526

Is Hardware Catching Up to Java?

The demands of parallel processing may be met more easily in a language we already know.

Nick Petreley, Editor in Chief

I had a wonderful experience chatting with the folks at RapidMind. The company provides C/C++ programmers with a very clever tool to help them exploit the parallel processing capability of multiple core CPUs. The idea is to make parallel processing at least somewhat transparent to the programmer. Programmers can use RapidMind to improve multicore performance of existing applications with minimal modifications to their code and build new multicore applications without having to deal with all of the complexities of things like thread synchronization.

RapidMind and any other solutions like it are invaluable and will remain so for a long, long time. C/C++ is an extremely well-entrenched legacy programming platform. I can't help but wonder, however, if the trend toward increased parallel processing in hardware will create a slow but steady shift toward inherently multithreaded, managed programming platforms like Java.

The inherent trade-off to adding cores to a CPU poses an interesting problem for any programming platform. As you add cores, you increase power consumption, which generates more heat. One way to reduce heat is to lower the processing speed of the individual cores, or at the very least, keep them from advancing in speed as quickly as they might have if history could have continued along its current path.

As a result, any single thread of execution would run faster on a speedy single-core CPU than it would on a slower core of a multicore CPU. The seemingly obvious answer is to split up the task into multiple threads. Java is multithreaded by nature, right? Therefore, all other things being equal, a Java application should be a natural for multicore CPU platforms, right?

Not necessarily. If Java was a slam-dunk superior way to exploit parallel processing, I would have posed this shift to Java as a prediction, not a question. It's not that simple. For example, don't let anyone tell you that Java was built from the ground up to exploit multiple processors and cores. It ain't so. Java's famous garbage collection got in the way of parallel programming at first. Java versions as late as 1.4 use a single-threaded garbage collector that stalls your Java program when it runs, no matter how many CPUs you may have.

But Java's multithreaded design more easily exploits parallel processing than many other languages, and it ended up lending itself to improvements in garbage collection. JDK 5.0 includes various tuning parameters that may minimize the impact of garbage collection on multiple cores or CPUs. It's not perfect, and you have to take into account the way your application is designed, but it is a vast improvement, and the improvement is made possible by the fact that Java was well conceived from the start.

Obviously, these features aren't enough. IBM builds a lot of additional parallelism into its WebSphere application server. In addition, IBM and other researchers are working on a Java-related language X10, which is designed specifically to exploit parallel processing (see **x10.sourceforge.net/x10home.shtml**).

It is also interesting to note that when Intel boasts about its quad-core performance on Java, its numbers are based on using the BEA jRockit JVM, not the Sun JVM. See **www.intel.com/performance/server/xeon/java.htm** for the chart and more information. I suspect Intel used this JVM because the BEA JVM employs garbage collection algorithms that are more efficient for use on multiple cores.

The fact that Intel used a non-Sun JVM makes this whole question of the future of Java on multicore CPUs interesting. I won't discount the possibility that Intel may have tuned its code to work best with the BEA JVM. But it is a significant plus for Java that you can choose a JVM best suited for the hardware you have. The big plus is that you still have to learn only one language, Java, and this does not limit your choice of architecture or platforms. If you run your application on a multicore platform, you can choose between JVMs and JVM-specific tuning parameters to squeeze extra performance out of your application.

Now, think a bit further into the future. Imagine parallelism showing up in a future generation of cell phones or other small devices. What better language than a platform-neutral one to take advantage of this future?

Some disclaimers are in order, though. First, I don't think any tool or language will soon make optimal programming for multiple cores or CPUs totally transparent. Java has the potential to exploit multiple cores with less impact to the programmer than many other languages, however, which is why I think Java's future is getting even more promising with this new hardware trend. Second, although I think Java has the edge, other managed, multithreaded languages will no doubt ride this same wave. Python, Ruby and your other favorite language in this genre probably have an equally bright future.■

Nicholas Petreley is Editor in Chief of *Linux Journal* and a former programmer, teacher, analyst and consultant who has been working with and writing about Linux for more than ten years.

FANATICAL SUPPORT™

Russ Barnard, President, FlapDaddy Productions

"Fanatical Support™ saved me from my own mistake."

"Not long ago, I reformatted one of our servers. Not until I was driving home did I learn that I brought our entire site down in the process. I called my guy at Rackspace and he said, 'We're already on it.' By the time I pulled in the driveway, my site was back up. Now that's Fanatical Support."

Keeping little mistakes from causing big problems is one definition of Fanatical Support. What will yours be?

Watch Russ's story at **www.rackspace.com/fanatical**
1-888-571-8976

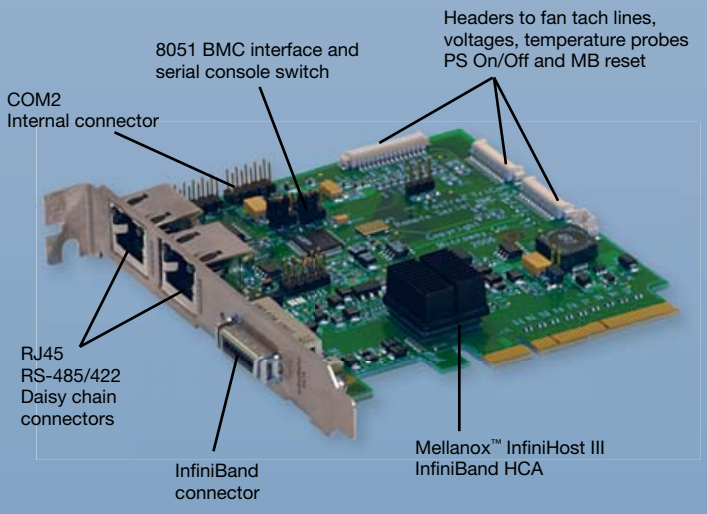
rackspace® MANAGED HOSTING

Affordable InfiniBand Solutions

4 Great Reasons to Call Microway NOW!

1 TriCom™

- DDR/SDR InfiniBand HCA
- "Switchless" serial console
- NodeWatch web enabled remote monitor and control



2 FasTree™

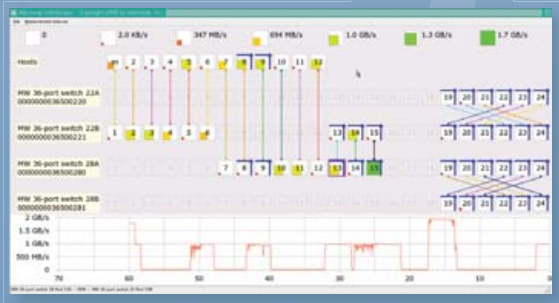
- DDR InfiniBand switches
- Low latency, modular design
- 24, 36 and 48 port building blocks

3 ServaStor™

- Extensible IB based storage building blocks
- Redundant and scalable
- Parallel file systems
- Open source software
- On-line capacity expansion
- RAID 0,1,1E, 3, 5, 6, 10, 50

4 InfiniScope™

- Monitors ports on HCA's and switches
- Provides real time BW diagnostics
- Finds switch and cable faults
- Lane 15 interface
- Logs all IB errors



Upgrade your current cluster, or let us design your next one using Microway InfiniBand Solutions.

To speak to an HPC expert call ***508 746-7341*** *and ask for technical sales or email*
sales@microway.com
www.microway.com

Microway®
Technology you can count on℠